UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

UNIVERSIDAD POLITÉCNICA DE VALENCIA

Campus de la Vera, Camino de la Vera, Valencia, España

MASTER EN AUTOMÁTICA E INFORMÁTICA INDUSTRIAL

SINTONÍA DE CONTROLADORES ROBUSTOS CON ALGORITMOS EVOLUTIVOS

Por

Pablo Cano Nistal

Tutor

Javier Sanchís Sáez

Contenido

Lista de figuras

Lista de tablas

Acrónimos

AG: Algoritmo Genético.

AE: Algoritmo Evolutivo.

CPOH: Control Predictivo y Optimización Heurística.

CPU: Unidad de Proceso Central.

DMC: Dynamic Matrix Control.

GA: Genetic Algorithm.

GPC: Control Predictivo Generalizado.

MIN: Mínimo.

MAX: Máximo.

MIMO: Multiple Input, Multiple Output.

MPC: Model Predictive Control.

QDMC: Quadratic Dynamic Matrix Control.

TS: Tiempo de Establecimiento.

Ts: Tiempo de muestreo.

TV: Total Variation:

SISO: Single Input, Single Output.

SO: Sistema Operativo.

SO: Sobreoscilación.

UPV: Universidad Politécnica de Valencia.

Agradecimientos

En primer lugar dar las gracias a mi tutor Javier Sanchís Sáez por ayudarme a lo largo de todo este año, las horas que me ha dedicado para ir encarrilando el trabajo y todas sus explicaciones.

También agradecer a la Universidad Politécnica de Valencia (UPV) el haberme formado durante este año y al grupo de Control Predictivo y Optimización Heurística (CPOH) del que he aprendido mucho y que me ha sido de gran ayuda para la realización de este trabajo.

Para terminar hacer una mención a mi familia y amigos que me han ayudado en todo lo que ha estado en su mano durante este año y en especial a Rubén y Adrián por soportar todas mis preguntas.

ESTA PÁGINA HA SIDO INTENCIONALMENTE DEJADA EN BLANCO

1. Introducción

En el presente capítulo se desarrollan las ideas que motivan este trabajo de fin de máster, enmarcada dentro del campo el control heurístico y que emplea técnicas de control predictivo.

1.1. Resumen

El objetivo del trabajo consiste en dotar de robustez a estrategias de control clásicas (tanto para sistemas de control SISO como MIMO) agregando dos filtros en el lazo de control: un filtro en la referencia y otro filtro en la realimentación.

Los parámetros de dichos filtros serán sintonizados utilizando algoritmos evolutivos, planteando un problema de minimización del peor caso (min-max). La función de coste será planteada como el tiempo de establecimiento del lazo cerrado sujeto a restricciones de sobreoscilación.

El trabajo limita la aplicación a procesos modelados como una función de transferencia de primer orden con retardo donde sus parámetros (ganancia, constante de tiempo y retardo) pueden variar dentro de un intervalo establecido.

Para el cálculo del peor caso se utilizará un algoritmo que calcula la envolvente de todas las respuestas de lazo cerrado y que engloba todas las respuestas de los posibles procesos que están dentro del intervalo definido. La metodología se aplicara a un ejemplo SISO controlado con un PID y a un ejemplo MIMO (3 x 3) controlado con un controlador predictivo multivariable DMC.

1.2. Motivación y objetivos

Como ya se menciona en el resumen, la motivación principal del trabajo es sintonizar una estructura de control basada en filtros (uno en la referencia y otro en la realimentación) para aumentar la robustez del sistema de control. Para ello, utilizaremos las ideas expuestas en los diferentes artículos citados en la Bibliografía y lo aplicaremos a sistemas SISO y MIMO.

Esta idea se ha basado en un sistema MIMO de una fábrica de papel (en la que se debe poder asegurar que se cumplirán todos los requisitos de calidad) controlada mediante control predictivo y robusto [1] y se ha simulado el proceso para ver las respuestas y compararlas con los datos aportados por los diferentes artículos. Este control se apoya en la idea de acelerar el proceso empleando únicamente los valores de los casos extremos de los parámetros que integran los procesos de primer orden que se van a controlar. Esta técnica, sumada a otras de ajuste multivariable y de control predictivo para el problema en cuestión, se lleva a cabo ignorando las restricciones del MPC y se simplifica en base a la estructura de un sistema de lazo cerrado.

Este tipo de sistema se basará en dos premisas a la hora de controlar los parámetros. La primera será mantener todos los parámetros fijos a excepción de uno respecto al que se realizará el ajuste y así poder simplificar el problema al conseguir un único grado de libertad. La segunda será ajustar los parámetros de forma que se pueda investigar la relación entre los mismos y el comportamiento del sistema en lazo cerrado.

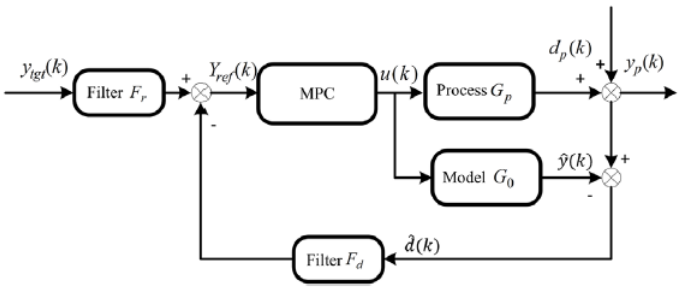

Ilustración 1.- Esquema del sistema [1]

Para comenzar, se plantea el sistema en base a la idea del artículo que se basa en la incorporación de filtros a la referencia y a la realimentación del sistema a controlar. Estos filtros contendrán un parámetro (λ) que servirá para modificar externamente el control que se realiza del sistema aumentando su robustez.

Este control en un primer momento se planteará con una perspectiva más tradicional, realizando los ensayos con un PID ajustado previamente para el proceso y posteriormente se sustituirá por controles más avanzado de tipo predictivo con y sin restricciones. Estos controles se ajustarán inicialmente y no serán modificados

internamente durante las simulaciones, pues ese cometido recae sobre los filtros que se han introducido previamente.

Además de los filtros, este sistema se diseña para evitar posibles errores (por tema perturbaciones, errores de modelado, etc.) aplicando un factor de corrección basado en la resta los casos extremos que se puedan encontrar a un modelo nominal previamente definido. Esta resta puede verse con mayor claridad en la Ilustración 2, pues en el artículo original [1] puede dar lugar a equívocos la forma en la que se lleva a cabo dicha corrección. Como se puede ver, la diferencia entre proceso nominal y real será la que es filtrada y luego adicionada de nuevo al modelo nominal para posteriormente realimentar a la entrada del proceso.

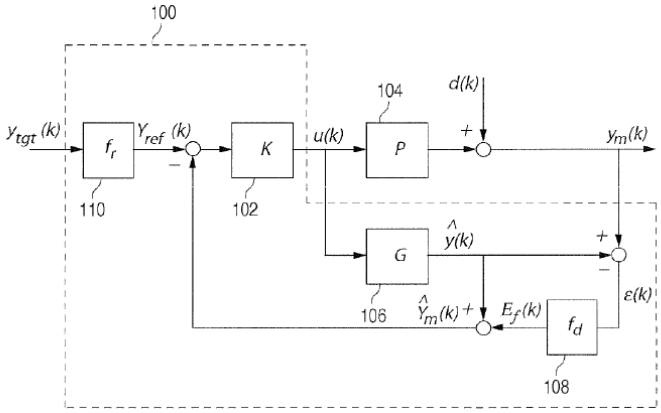

Ilustración 2.- Esquema detallado del sistema.

El filtro que se introduce a la entrada justo a continuación de la referencia, tendrá como objetivo suavizar dicha referencia según el parámetro λ del filtro en cuestión.

En los ensayos en los que se implementaron técnicas de control predictivo, se fue realizando el ajuste de los parámetros y ajustando en función de los resultados obtenidos. Se debe prestar especial atención en las dimensiones de las variables utilizadas al ser uno de los mayores focos de problemas al extender al caso MIMO. Se probó tanto la técnica de control predictivo sin restricciones DMC (véase 2.3.2), como el caso que si las incorpora QDMC (véase 2.3.3) y ambos casos permiten una respuesta interesante en comparación a la obtenida por los diferentes artículos consultados.

Para llevar a cabo todo lo anteriormente comentado, también se hará uso de algoritmos genéticos como herramienta para la optimización de funciones de coste. La solución J en los ensayos iniciales es directamente el tiempo de establecimiento (que

es el elemento que se desea optimizar). Además, se han implementado restricciones [6] sobre la sobreoscilación y sobre la variación total de forma que se penaliza un valor por encima del límite fijado para un proceso concreto que puede ser modificado programáticamente. Esta penalización se genera de dos posibles formas:

- Incorporando un valor de compensación que se añada a la solución J de la función de coste en caso de no cumplir los requisitos especificados:

$$J = J + offset$$

- Incorporando un valor de compensación que dependa del grado de violación del requisito especificado. Este valor de violación se obtendrá como valor en porcentaje del parámetro y permitirá además una mejor selección entre las opciones que sí que cumplan los requisitos:

$$violation = (parámetro - valor\_máx\_parámetro) / valor\_máx\_parámetro$$
$$J = J + violation \cdot offset$$

Este proceso iterativo devolverá una solución después de que especifiquemos una serie de parámetros necesarios para el correcto funcionamiento del algoritmo como el número de individuos por cada generación, el número de generaciones, los limites inferior y superior de los parámetros a optimizar (que en el caso de este proyecto serán los parámetros de los filtros).

Una vez obtenida una solución óptima del proceso mediante los AG y que ya dispone de un control interno, bien PID o bien de tipo predictivo, se prosigue con la obtención de una envolvente o sobre que abarque todas las posibles soluciones y que esté basado en los valores extremos de los parámetros del proceso real y nominal.

Se confirma la efectividad de este método al visualizar gráficamente empleando únicamente los valores de los extremos y luego verificando la obtención de una envolvente pareja a la anterior pero incorporando todos los posibles valores que pudiesen estar incluidos dentro de esos peores casos. Para ello se van almacenando los valores máximos y mínimos de los procesos simulados y a continuación se realiza un simple ploteado que permita superponer las gráficas de ambos procesos y comparar los resultados.

ESTA PÁGINA HA SIDO INTENCIONALMENTE DEJADA EN BLANCO

2. Metodología y herramientas

2.1. Algoritmos Genéticos

Las primeras referencias a los algoritmos evolutivos las encontramos a finales de los cincuenta cuando el biólogo inglés Alex S. Fraser publicó un conjunto de trabajos [2] sobre sistemas biológicos que sirvieron de inspiración para los algoritmos genéticos. Más tarde, en la década de los sesenta, John Holland propuso el uso de mecanismos evolutivos como forma de optimización dentro del campo de la inteligencia artificial. [3]

Gracias a la aparición de ordenadores más potentes y baratos, fue a partir de la década de los 80s cuando comenzaron a cobrar importancia y demostrar su potencial.

Existen algunos problemas reales que pueden surgir tanto en ingeniería como en otros ambientes que no pueden ser resueltos con métodos tradicionales y para los cuales una de las posibles aproximaciones para solucionarlos es la aplicación de algoritmos evolutivos (AE), los cuales permiten explorar un amplio rango de soluciones potenciales.

Estos problemas pueden formularse como problemas de optimización y están inspirados en modelos evolutivos presentes en la naturaleza, basados en la selección natural, que establece los individuos más aptos, y en la reproducción sexual, que verifica diversidad a la población, a través de cambios continuos y aleatorios de la misma.

Dentro de los algoritmos evolutivos, los algoritmos genéticos (AG) son uno de los tipos más extendidos y los principales procesos de transformación que pueden sufrir las distintas poblaciones para producir una nueva generación son:

- Selección de los individuos: Elección basada en la aptitud de cada individuo, en la que se evalúa y selecciona una cantidad finita de individuos que se cruzarán entre sí. Además, se debe especificar una regla de aptitud para realizar la selección de que candidatos de cada población pasan a la siguiente generación.
- Cruce: Intercambio de características (cromosomas) entre dos individuos que darán lugar a dos descendientes.

- Mutación: De forma similar a los procesos biológicos, se añade un parámetro de mutación para alcanzar y aumentar el espacio de búsqueda de la población actual y que aporten diversidad a la solución.

El proceso del AG es iterativo. Esto quiere decir que para un grupo de población, se le someterá a los procesos descritos arriba generando un par descendiente por cada par de individuos. Siempre se escoge al más apto (dentro de lo posible). El algoritmo repetirá el proceso para la nueva generación, terminando cuando el usuario elija el criterio de finalización deseado. Se debe tener en cuenta que a mayor número de individuos y de generaciones, es más fácil que se obtengan mejores resultados, aunque hay que tener cuidado con posibles mínimos locales a los que converja la función.

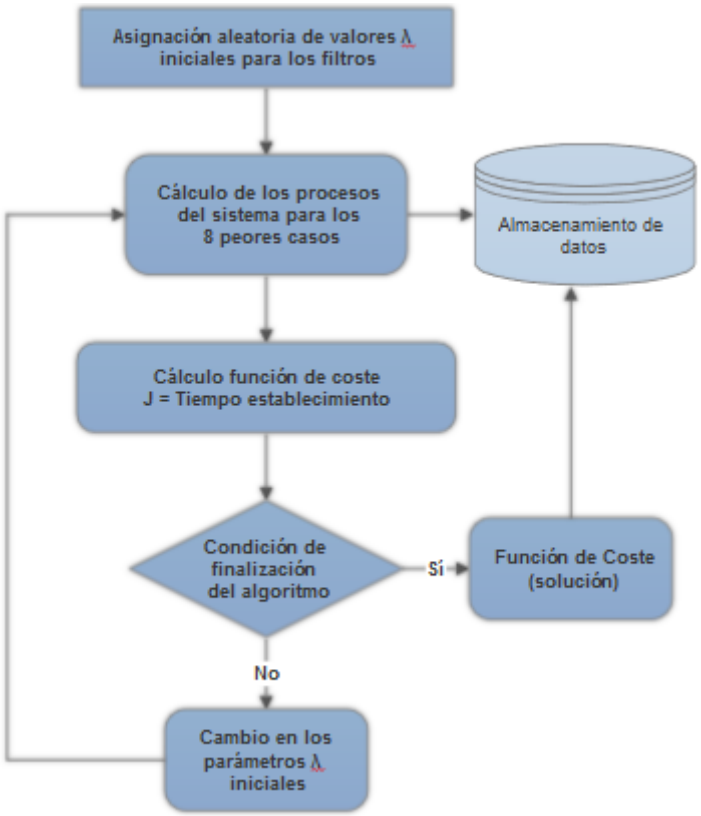

Ilustración 3.- Flujograma algoritmo evolutivo

Como se puede ver en el flujograma (Ilustración 3), el algoritmo evolutivo irá calculando todos los sistemas reales posibles y los filtros que cambian debido a la modificación de los valores λ (véanse los bloques coloreados de la Ilustración 4)

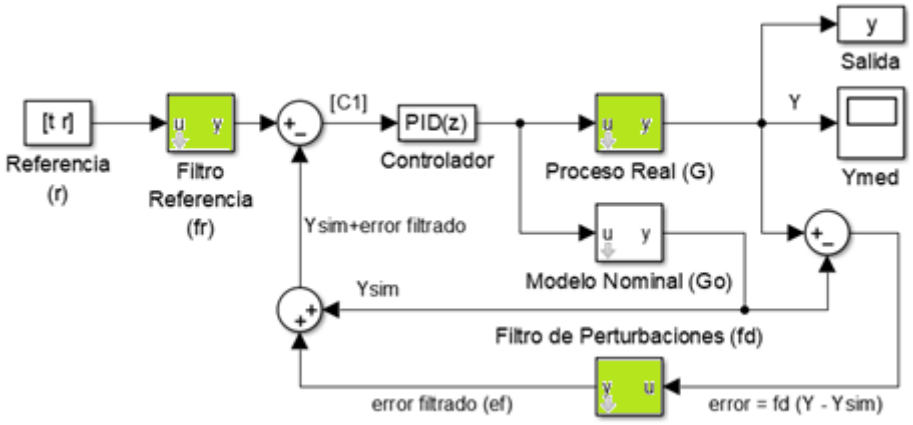

Ilustración 4.- Esquema de los bloques a los que afecta el algoritmo evolutivo

2.1.1. Basic Genetic Algorithm

La versión del algoritmo genético utilizado en este trabajo ha sido desarrollado y pertenece al grupo de Control Predictivo y Optimización Heurística (CPOH) de la Universidad Politécnica de Valencia [5].

Esta versión simplifica en gran medida el uso de algoritmos genéticos gracias a una combinación ya definida de cruces, mutaciones, etc. La efectividad del algoritmo empleado ha probado ser de gran valor y de ahí su implementación en este proyecto.

2.1.2. Algoritmo genético con restricciones

Desde que se ha generalizado el uso de los algoritmos genéticos como métodos de búsqueda de soluciones óptimas, se ha empleado la función de penalización (penalti) en la mayoría de los casos [6]. Esta función es muy útil a la hora de añadir restricciones a un sistema como el planteado que seguirá el flujograma que se puede encontrar en la Ilustración 5.

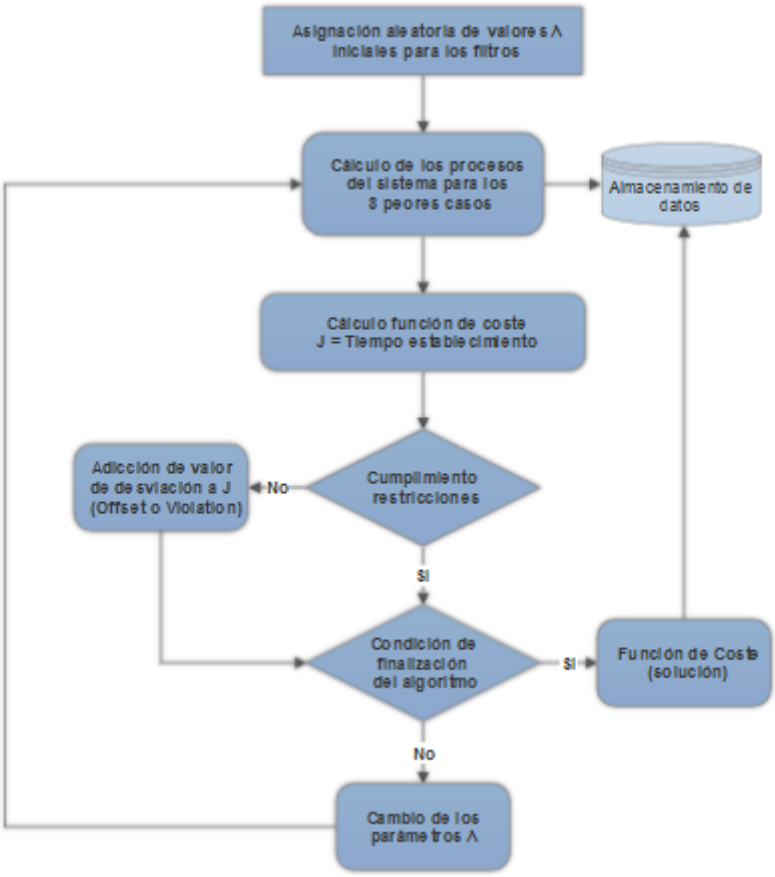

Ilustración 5.- Flujograma algoritmo evolutivo con restricciones

2.2. Control tradicional PID

El tipo de control más extendido para controlar cualquier tipo de proceso es el PID. Esto es debido a su buena adaptabilidad y la sencilla modificación de sus parámetros, todo ello con un rendimiento bastante aceptable.

A pesar de su popularidad y de sus ventajas, los controles PID tienen un rango de actuación limitado al seguimiento de una referencia y pueden tener problemas a la hora de controlar procesos complejos y no lineales (algo que suele ser bastante habitual).

2.3. Control Predictivo

En los setenta surgieron algoritmos que mediante un modelo dinámico del proceso predecían los efectos de las acciones de control futuras en la salida, minimizando el error predicho en base a restricciones. Estas formulaciones de naturaleza heurística y algorítmica, buscaban aprovechar el potencial de los ordenadores y fueron el inicio de lo que más tarde se denominaría control predictivo (MPC). Es más, en la industria se implementaron controles predictivos varios años antes de aparecer las primeras publicaciones, basándose en algoritmos de control sencillos, con restricciones de funcionamiento incorporadas en la formulación del problema. Posteriormente se fue dando fundamentos teóricos desde universidades y otros centros de investigación.

La popularidad del MPC creció rápidamente en las industrias con procesos químicos por ser bastante intuitivo y debido sobre todo a la simplicidad del algoritmo y a la facilidad para identificar.

Algunos de los algoritmos de control predictivo empleados a lo largo de la historia han sido GPC, DMC, QDMC, etc. A lo largo de esta tesina se han empleado los dos últimos de los mencionados.

Dentro del MPC industrial, las principales limitaciones que se encuentran son que se tienen modelos sobreparametrizados con modelos empíricos obtenidos en base a experimentos y en los que no se suele usar modelo de las perturbaciones.

Según avanzan los años y evolucionan los procesos, en la industria se ha vuelto necesario que el control de los procesos cumpla unos requisitos (económicos, de calidad, de seguridad, medioambientales, etc.) cada vez más avanzados para satisfacer y predecir los cambios y la competencia del mercado.

Para poder conseguir todas estas especificaciones, se dispone de varios tipos de metodologías de control, pero a lo largo de este trabajo de fin de máster se le dará mayor importancia al control predictivo basado en modelos (MPC), debido a que acepta cualquier tipo de modelo, funciones objetivo y restricciones.

El control predictivo podría explicarse con sencillez si se compara con la conducción de un coche, en la que un conductor va realizando modificaciones en función de la carretera, por ejemplo, frenar un poco antes de entrar en una curva o acelerar si se aproxima a una recta prolongada. Todo ello con un objetivo final que podría ser llegar antes al destino, reducir el consumo de combustible, etc.

2.3.1. MPC

El control predictivo basado en modelos (MPC) es aquel que requiere de un modelo dinámico del proceso que bien puede ser analítico o empírico. Para obtener unos objetivos que vienen definidos bajo el nombre de función de coste, se hace uso de un optimizador que se encargará de encontrar la mejor secuencia de acciones de control para alcanzar dicho objetivo propuesto.

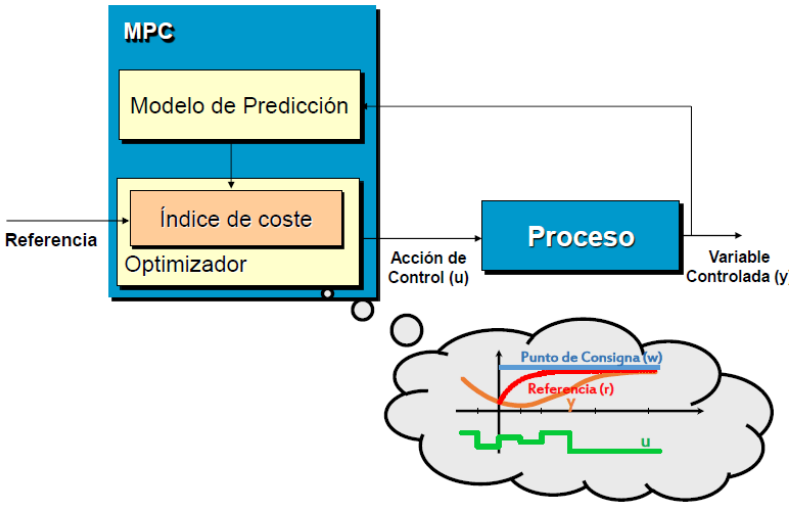

Ilustración 6.- Esquema MPC. Asignatura Control Inteligente y Predictivo

Se toma como ejemplo teórico de función de coste J a (1) y las restricciones (2) y (3). Se tendrán una serie de parámetros de sintonía del controlador $\varphi, \lambda, P\ y\ M$ unas predicciones y(k+i|k), unas referencias futuras $y_{sp}(k+i)$ y unos controles futuros $\Delta u\ (k+i)^2$.

$$\min J_{\Delta u(k+i)} = \sum_{i=1}^{P} \varphi(i)\ [y(k+i|k) - y_{sp}(k+i)]^2 + \sum_{i=0}^{M-1} \lambda(i+1)\ \Delta u\ (k+i)^2 \qquad \textbf{(1)}$$

Sujeto a $u_{min} < u(k+i) < u_{max}$ **(2)**

$Y_{min} < y(k+i) < y_{max}$ **(3)**

Para explicar el control predictivo que se va a emplear en este proyecto se debe entender el concepto de modelos de convolución. En este tipo de modelo, la predicción de la salida se consigue como la suma de la respuesta libre (efecto de las

entradas pasadas si las futuras fuesen nulas) y de la respuesta forzada (efecto de la entrada actual y las futuras, considerando que el sistema parte del reposo), véase Ilustración 7.

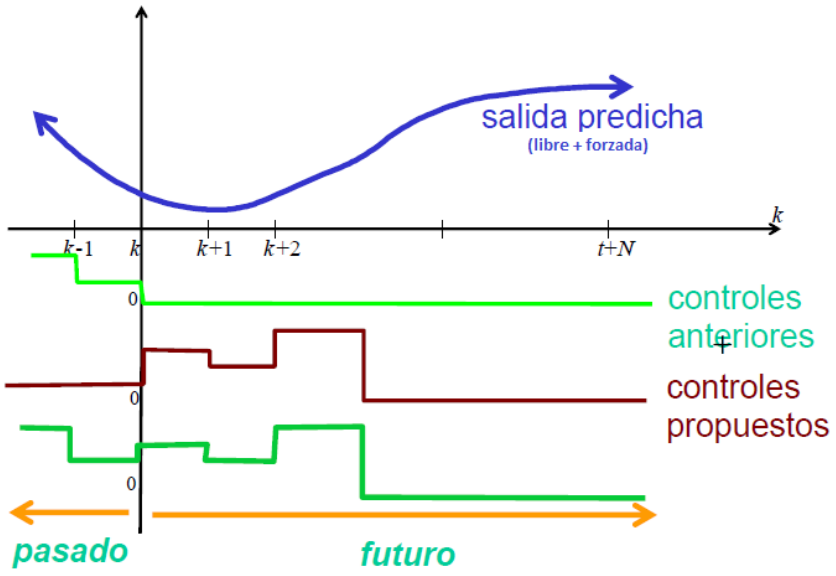

Ilustración 7.- Modelo de convolución

2.3.2. Dynamic Matrix Control (DMC)

El DMC es un tipo de control predictivo en el que se asume que tanto las perturbaciones medibles d(k) como las no medibles e(k) se mantienen constantes en el futuro:

$$d(k) = d(k+1|k) = d(k+2|k) = \ldots = d(k+p-1|k) \quad \textbf{(4)}$$

$$\Delta d(k|k) = d(k) - d(k-1) = 0 \quad \textbf{(5)}$$

$$e(k) = e(k+1|k) = \ldots = e(k+p|k) \quad \textbf{(6)}$$

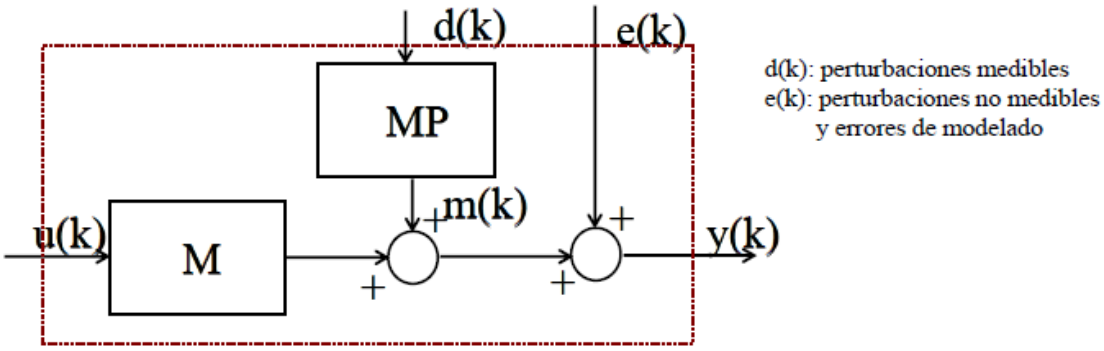

Ilustración 8.- Esquema DMC

$$\begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ y(k+p|k) \end{bmatrix} = \begin{bmatrix} f(k+1|k) \\ f(k+2|k) \\ \vdots \\ f(k+p|k) \end{bmatrix} + \begin{bmatrix} s^u(1) & 0 & 0 & 0 \\ s^u(2) & s^u(1) & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s^u(p) & s^u(p-1) & \cdots & s^u(1) \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+p-1|k) \end{bmatrix} + \begin{bmatrix} s^d(1) \\ s^d(2) \\ \vdots \\ s^d(p) \end{bmatrix} \cdot \Delta d(k|k) + \begin{bmatrix} y_m(k) - f(k|k) \\ y_m(k) - f(k|k) \\ \vdots \\ y_m(k) - f(k|k) \end{bmatrix}$$

Ilustración 9.- Expresión del DMC

Se podrá simplificar la expresión que se encuentra en la Ilustración 9 como se ve en (7) para obtener una más compacta (8).

$$Y = G \cdot \hat{u} + f + S^d \cdot \Delta d + e \qquad \textbf{(7)}$$

$$Y = G \cdot \Delta u + L \qquad \textbf{(8)}$$

2.3.3. QDMC (Quadratic Dynamic Matrix Control)

Las versiones de los algoritmos de control predictivo utilizados en este trabajo han sido desarrolladas y pertenecen al grupo de Control Predictivo y Optimización Heurística (CPOH) de la Universidad Politécnica de Valencia [5]. Se han utilizado después del aprendizaje y prueba de los mismos en una asignatura del máster de automática e informática industrial.

El QDMC incorpora las restricciones a la base del control DMC. Estas restricciones pueden ser sobre las acciones de control (10) (11) o sobre las salidas (12). Esto generará una función de coste de la forma (9):

$$min_{\Delta u}\, J = min_{\Delta u}\, [\, (G\, \Delta \mathrm{U} - \mathrm{E})^{\mathrm{T}}\, \varphi\, (\mathrm{G}\, \Delta \mathrm{U} - \mathrm{E}) + \Delta \mathrm{U}^{\mathrm{T}}\, \lambda\, \Delta \mathrm{U}\,] \qquad \textbf{(9)}$$

$$\Delta \mathrm{U}_{min} \leq \Delta \mathrm{U} \leq \Delta \mathrm{U}_{\max} \qquad \textbf{(10)}$$

$$\mathrm{U}_{min} \leq \mathrm{U} \leq \mathrm{U}_{\max} \qquad \textbf{(11)}$$

$$\mathrm{Y}_{min} \leq \mathrm{Y} \leq \mathrm{Y}_{\max} \qquad \textbf{(12)}$$

De esta forma la expresión en la que se incorporan las restricciones será (véase Ilustración 10):

$$\begin{bmatrix} I \\ -I \\ I_L \\ -I_L \\ G \\ -G \end{bmatrix} \Delta U \leq \begin{bmatrix} \Delta u_{\max_1} \\ \Delta u_{\max_2} \\ \vdots \\ \Delta u_{\max_c} \\ -\Delta u_{\min_1} \\ -\Delta u_{\min_2} \\ \vdots \\ -\Delta u_{\min_c} \\ u_{\max_1} - u(k-1) \\ u_{\max_2} - u(k-1) \\ \vdots \\ u_{\max_c} - u(k-1) \\ -u_{\min_1} + u(k-1) \\ -u_{\min_2} + u(k-1) \\ \vdots \\ -u_{\min_c} + u(k-1) \\ Y_{\max} - L \\ -Y_{\min} + L \end{bmatrix}$$

Ilustración 10.- Expresión QDMC

ESTA PÁGINA HA SIDO INTENCIONALMENTE DEJADA EN BLANCO

3. Ensayos y análisis de resultados

3.1. Control PID de un SISO: obtención envolvente con Simulink

Considerando un proceso estable simple entrada simple salida (SISO), se toma como modelo de prueba el modelo de primer orden (13) (P en la Ilustración 11):

$$Gp(s) = \frac{\mathrm{K}}{\tau\, s+1}\, e^{-Ts} \qquad \textbf{(13)}$$

Para este modelo, se toma un rango de valores para cada uno de los parámetros K, T y τ y se toma el valor medio como valor para el modelo nominal:

Tabla 1.- Parámetros del modelo de prueba y sus posibles valores

Parámetro	Valor Mínimo	Valor Máximo	Valor Nominal
K	1	3	2
T	1	3.5	2.25
T	7	12	9.5

Del mismo modo, se debe definir un modelo nominal (14), que se puede ver definido como G en Ilustración 11 que sigue la forma:

$$Gn(s) = \frac{\mathrm{K}_0}{\tau_0\, s+1}\, e^{-T_0 s} \qquad \textbf{(14)}$$

El esquema que se puede ver en la Ilustración 11, se ha reproducido mediante Simulink para poder simular el comportamiento del sistema ante el modelo definido. Este esquema, permite corregir el error entre el proceso real y el modelo del sistema. Además, se tienen dos filtros (15) y (16), el primero para llevar a cabo un filtrado de la

referencia y el segundo para el error antes mencionado. Estos filtros poseen la forma [7]:

$$fd(s) = \frac{1}{(\tau\lambda_d)\,s+1}\, e^{-Ts} \qquad \textbf{(15)}$$

$$fr(s) = \frac{1}{(\tau\lambda_r)\,s+1}\, e^{-Ts} \qquad \textbf{(16)}$$

Con estos filtros se pretende facilitar el control de la planta, fijando los parámetros del controlador para solo tener que ajustar los valores λ_d, λ_r de los filtros f_d y f_r.

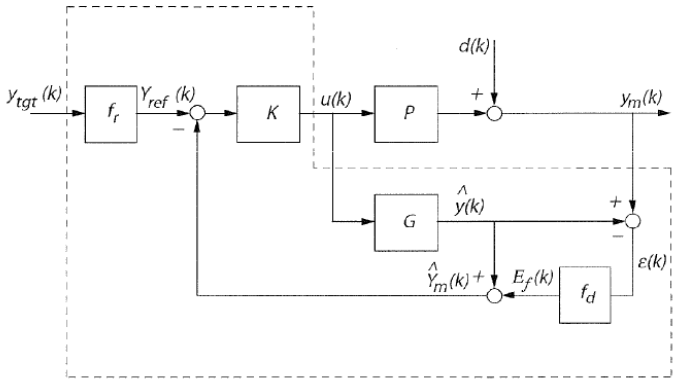

Ilustración 11.- Esquema del sistema [2]

Si para empezar se toma un valor nulo para los valores λ_d, λ_r, se obtiene un resultado con bastante sobreoscilación y poco eficaz (véase Ilustración 12).

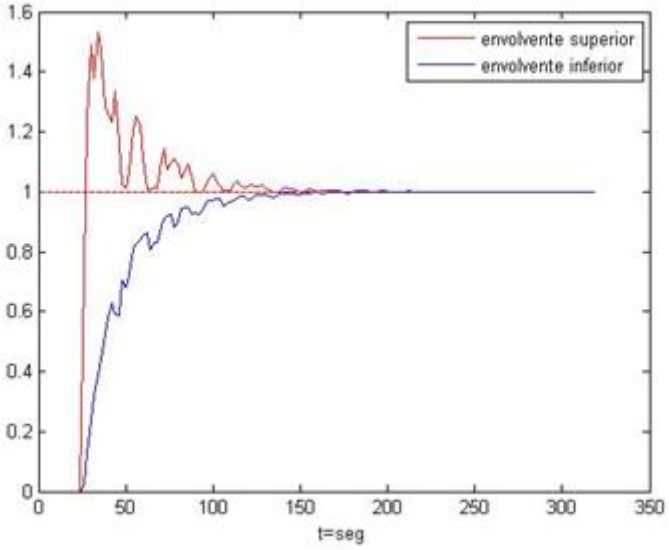

Ilustración 12.- Envolvente con valores de landas nulos

Si por el contrario se toma un valor cualquiera aleatorio, el resultado que se obtiene consigue un resultado bastante distinto, con el que se ve la importancia de un ajuste óptimo de estos parámetros (véase Ilustración 13).

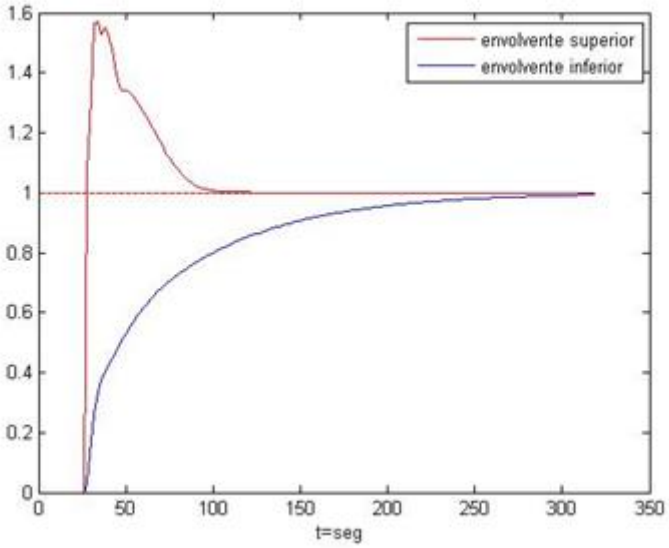

Ilustración 13.- Envolvente con valores aleatorios de los parámetros landas

Para terminar, se toman los valores óptimos calculados con un algoritmo genético (véase Tabla 4) del que más adelante en el trabajo se hablará con más profundidad y como resultado se obtiene una mejora considerable en la envolvente calculada (véase Ilustración 14).

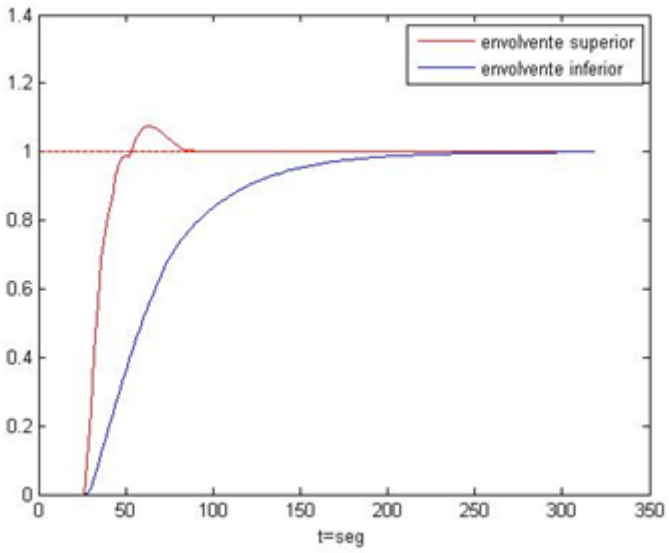

Ilustración 14.- Envolvente con valores de landas calculados por el AG

Para conseguir las envolventes del sistema se ejecuta el código que se ha desarrollado en Matlab [8] (véase Anexo 1.C). Este código dispone de una llamada a Simulink que posteriormente se ha eliminado con el fin de reducir los tiempos de cómputo de la simulación (véanse anexos 1.C y 1.D).

Con el fin de facilitar la comprensión del sistema, se buscará modificar la disposición de los bloques de forma que queda una configuración típica de control en lazo cerrado, con un controlador, el proceso real y la realimentación. Para ello, se tiene que alcanzar la configuración previa al cambio (Ilustración 15):

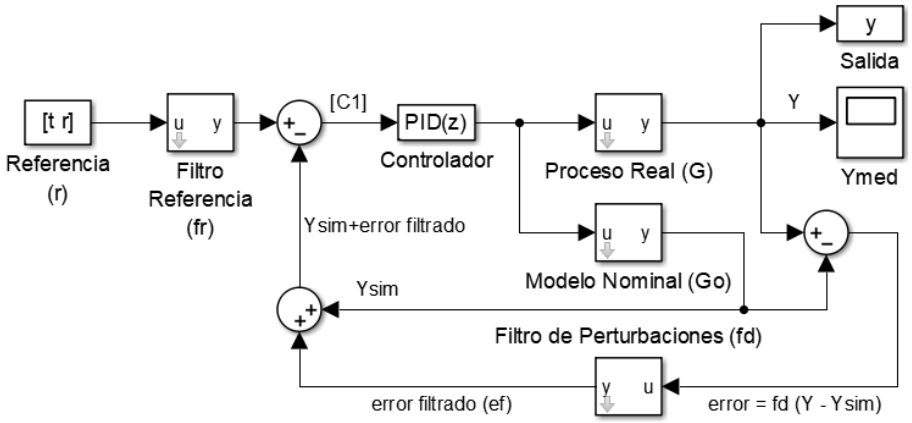

Ilustración 15.- Esquema del sistema en Simulink

$$[C1]: \quad r \cdot fr - (ef + Ysim)$$

$$ef = fd \cdot (Y - Ysim) = fd \cdot Y - fd \cdot Ysim$$

$$[\boldsymbol{C1}]: \quad \boldsymbol{r \cdot fr - (fd \cdot Y - fd \cdot Ysim + Ysim)}$$

Como se puede observar, la envolvente de las respuestas obtenidas por los modelos que emplean únicamente los valores extremos es la misma que si se calcula utilizando todos los valores intermedios (Véanse Ilustración 16 e Ilustración 17). Esto permite ahorrarse tiempo de cómputo, lo que facilitará que el control del sistema sea más rápido.

Tabla 2.- Diferencias de tiempo de cómputo CPU

Tiempo de cómputo (segundos)	**Todos los valores intermedios** ***(***Ilustración 16***)***	**Valores extremos** ***(***Ilustración 17***)***
Ensayo 1	28,796875	2,375000
Ensayo 2	27,656250	2,218750
Ensayo 3	27,875000	2,250000
Ensayo 4	36,609375	2,234375
Ensayo 5	36,218750	2,140625
Ensayo 6	27,468750	2,218750

Ensayo 7	26,062500	2,218750
Ensayo 8	26,125000	2,187500
Ensayo 9	25,859375	2,187500
Ensayo 10	25,609375	2,203125
Media	**28.828125**	**2.223438**

Como se puede observar en la Tabla 2, la reducción del tiempo de cómputo es bastante notoria.

Para mostrar en una gráfica todos los valores intermedios se seleccionarán los vectores:

$$K = (1, 2, 3)$$

$$Tau = (7, 8, 9, 10, 11, 12)$$

$$T = (1, 1.5, 2, 2.5, 3, 3.5)$$

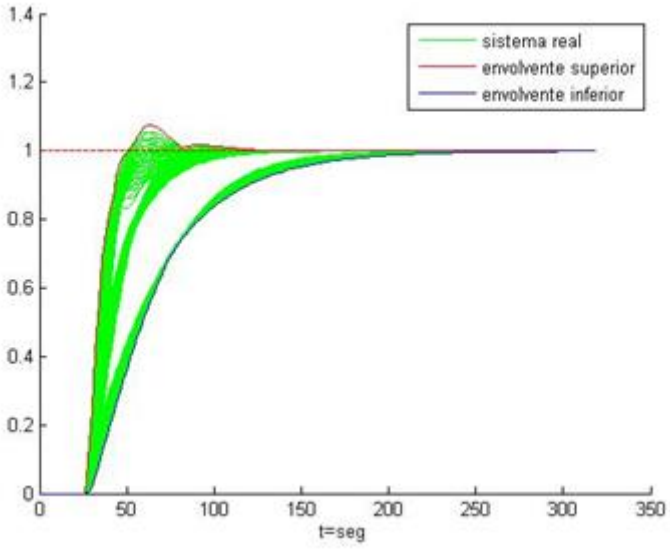

Ilustración 16.- Envolventes de los modelos dentro del rango de valores extremos

Para mostrar en una gráfica únicamente los valores extremos se seleccionarán los vectores:

$$K \in [1, 3]$$

$$Tau \in [7, 12]$$

$$T \in [1, 3.5]$$

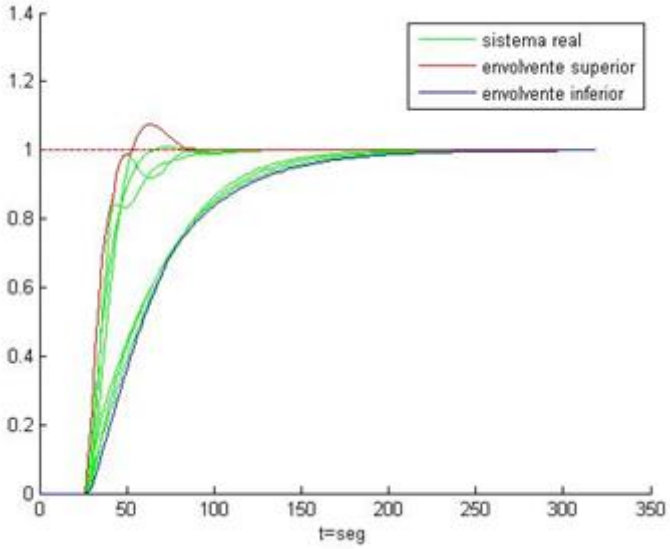

Ilustración 17.- Envolventes de los modelos de los valores extremos

Entre estos dos casos se puede verificar que se obtiene la misma envolvente al realizar el cálculo con valores extremos y con todos los valores (véase Ilustración 18).

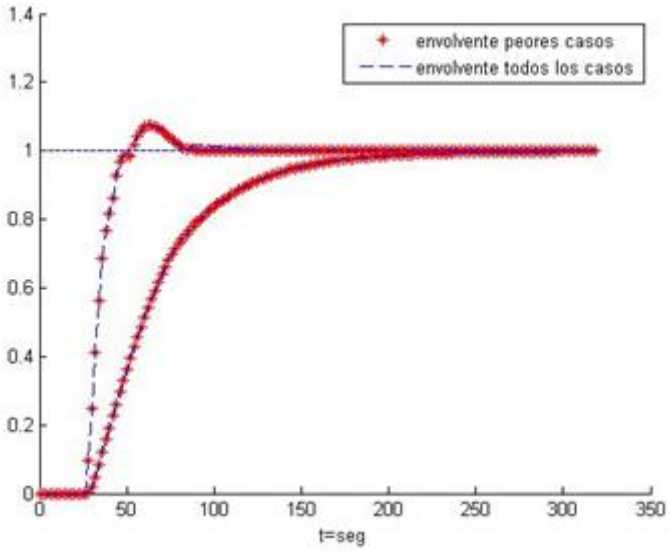

Ilustración 18.- Comparativa de envolventes

3.2. Control PID de un SISO: obtención envolvente sin Simulink

Sabiendo que la configuración en el punto previo al controlador debe ser [C1], se puede reubicar los bloques de forma que se consiga un conjunto de bloques más similar al buscado (Ilustración 19):

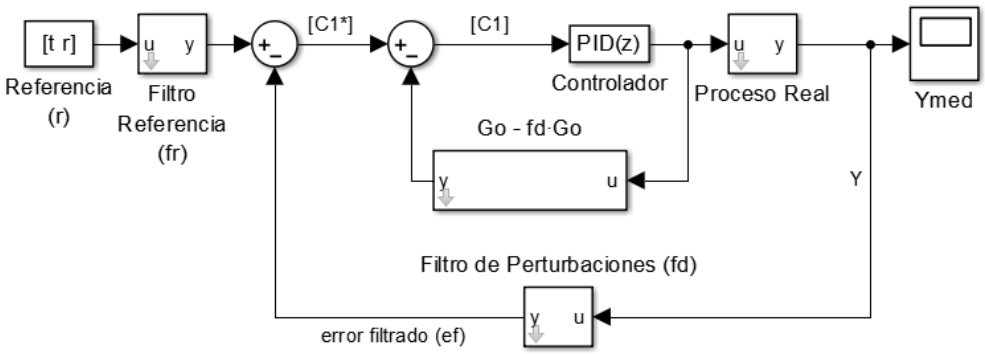

Ilustración 19.- Esquema modificado del sistema en Simulink

$$[C1^*]: \quad r \cdot fr - fd \cdot Y$$

$$Ysim = PID \cdot Go$$

$$Go - fd \cdot Go = Ysim - fd \cdot Ysim$$

$$\boldsymbol{[C1]: \quad r \cdot fr - fd \cdot Y + fd \cdot Ysim - Ysim}$$

Con estos cambios, el software desarrollado en Matlab se ve reducido, bajando notoriamente el coste computacional, como se puede ver en la Tabla 3.

Tabla 3.- Comparador entre tiempos de cómputo con y sin Simulink

Tiempo de computo (segundos)	**Sin Simulink**	**Con Simulink**
Ensayo 1	1,796875	2,375000
Ensayo 2	0,656250	2,218750
Ensayo 3	0,671875	2,250000
Ensayo 4	0,718750	2,234375
Ensayo 5	0,640625	2,140625
Ensayo 6	0,671875	2,218750
Ensayo 7	0,687500	2,218750
Ensayo 8	0,734375	2,187500
Ensayo 9	0,828125	2,187500
Ensayo 10	0,656250	2,203125
Media	**0.806250**	**2.223438**

Viendo los resultados obtenidos se opta por continuar las simulaciones del caso SISO sin simulink.

Para comprobar que todas las soluciones previstas están recogidas dentro de esta envolvente, se pueden probar diferentes modelos, asignando valores a los parámetros que estén dentro del rango asignado. Si se muestra en una gráfica las respuestas de todos los posibles modelos frente a la envolvente generada por el modelo basado en los extremos, se puede observar que no se aprecian diferencias (véase Ilustración 20).

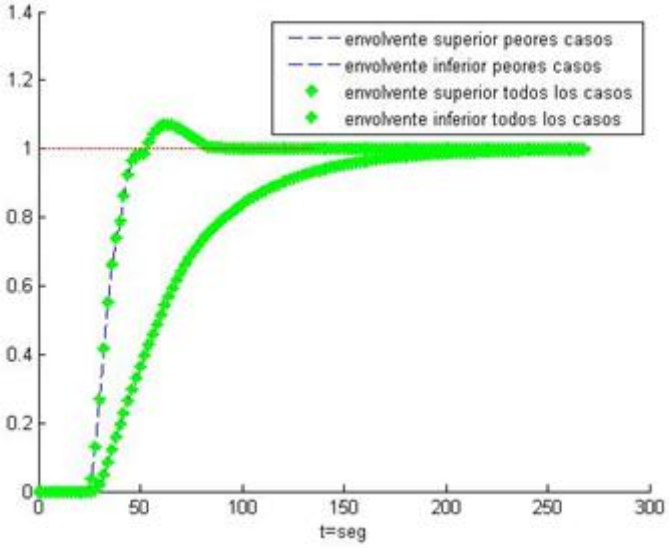

Ilustración 20.- Comparativa envolventes sin simulink

Se puede observar que la envolvente obtenida mediante los peores casos abarca las respuestas de los posibles sistemas intermedios [3].

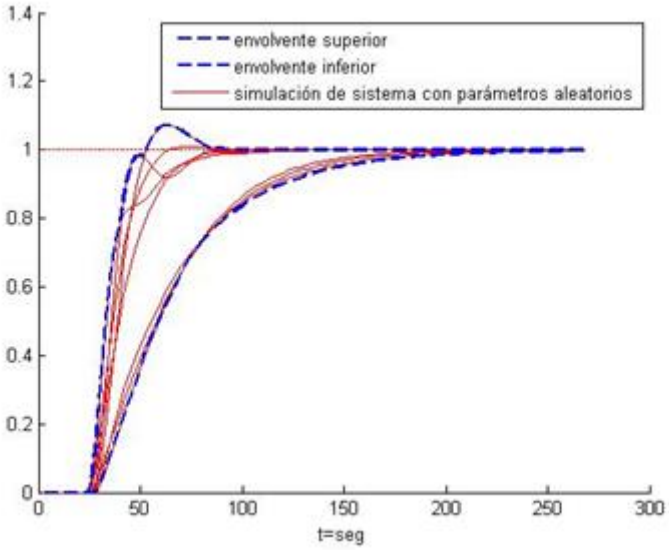

Ilustración 21.- Respuesta a sistema intermedio aleatorio.

3.3. Sintonía del sistema de control con algoritmos genéticos

Se han utilizado los algoritmos genéticos para obtener el mínimo de la función objetivo. Para esto primero se ha desarrollado una función objetivo (ver anexos 1.B, 1.D, 1.F, 1.H y 1.J) en la que se calcula el tiempo de establecimiento (TS), sobreoscilación (SO) y variación total (TV). Posteriormente esta función se añade al código principal dentro del algoritmo genético junto con distintos parámetros para tratar de optimizar el resultado obtenido. Para poder arrancar el algoritmo, son necesarios algunos parámetros:

- Nombre de la función objetivo.
- Número de variables a optimizar (nvars): En este caso, los valores que se desea optimizar son los parámetros de los filtros (λ_d y λ_r). Este parámetro solo es necesario para el uso del algoritmo genético de Matlab, puesto que en el BasicGA no es necesario, pues ya viene definido por defecto.
- Número de individuos de la población.
- Límites: Se deberán escoger unos límites que responderán a las siguientes inecuaciones (17):

$$0 \leq \lambda_d \leq 3 \qquad\qquad 0 \leq \lambda_r \leq 3$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} \lambda_d \\ \lambda_r \end{pmatrix} \leq \begin{pmatrix} 3 \\ 3 \\ 0 \\ 0 \end{pmatrix} \qquad \textbf{(17)}$$

Si se ejecuta el BasicGA, indicando los parámetros antes citados, cuando termine todas las iteraciones que estén estipuladas se obtendrá un resultado, que será mostrado en pantalla a través de un mensaje del estilo:

RESULT

Objective function for xmin: 296

xmin: [2.74168328140786 1.33739296654269]

Para el caso que se trata en el trabajo, la función objetivo (18) que tratará de minimizar el algoritmo será el tiempo de establecimiento (o la suma de los tiempos de establecimiento si se trata de MIMO):

$$J = \min \sum_{n=1}^{\infty} (\text{Tiempo\_Establecimiento}) \qquad \textbf{(18)}$$

Se puede probar a disminuir el tamaño de la población, lo que reducirá el tiempo y se obtendrán unos resultados que se pueden ver en las Tabla 4 y Tabla 5.

Tabla 4.- Resultados BasicGA para población de 50 individuos y 50 iteraciones

Prueba	**λ_d**	**λ_r**	**Tiempo de establecimiento**	**Tiempo Cómputo**
Ensayo 1	0,4976	0,0010	44,4031	840,7344
Ensayo 2	1,0191	1,4992	39,2785	836,8281
Ensayo 3	1,0191	1,4993	39,2793	840,5000
Ensayo 4	1,0202	1,5023	39,2930	838,2500
Ensayo 5	1,0192	1,4995	39,2802	916,5625
Ensayo 6	1,0329	1,5401	39,4488	840,6563
Ensayo 7	1,0191	1,4992	39,2787	837,6406
Ensayo 8	1,0192	1,4992	39,2801	838,8125
Ensayo 9	1,0191	1,4992	39,2787	838,4219
Ensayo 10	1,0365	1,5508	39,4928	864,0781
Min	***1,0191***	***1,4992***	**39,2785**	**836,8281**
Max	***0,4976***	***0,0010***	**44,4031**	**916,5625**
Media			**39,8313**	**849,2484**

Tabla 5.- Resultados BasicGA para población de 10 individuos y 50 iteraciones

Prueba	λ_d	λ_r	Tiempo de establecimiento	Tiempo Cómputo
Ensayo 1	0,7129	1,5933	48,7760	722,4531
Ensayo 2	0,6764	1,4603	47,6092	765,5000
Ensayo 3	0,3988	0,0693	50,8826	696,8594
Ensayo 4	0,4715	0,1672	52,9853	594,9688
Ensayo 5	0,6765	1,4609	47,6120	594,7500
Ensayo 6	0,6827	1,4830	47,7712	598,4531
Ensayo 7	0,6673	1,4221	47,3874	594,2344
Ensayo 8	0,0000	1,7511	50,0065	595,5469
Ensayo 9	0,0000	1,7511	50,0063	595,6250
Ensayo 10	0,9359	2,1300	57,1764	595,5625
Min	***0,6673***	***1,4221***	**47,3874**	**594,2344**
Max	***0,9359***	***2,1300***	**57,1764**	**595,5625**
Media			**50,0213**	**635,3953**

Una vez conseguida una función de coste f(x) que minimice el tiempo de establecimiento hasta alcanzar un valor óptimo F(x), se procede a añadir restricciones sobre la sobreoscilación y sobre la variación total. Para hacerlo, se añaden unos valores que penalicen en función del valor. Se calculará el porcentaje de SO (19) y TV (20) para aplicar la penalización (violation) en función de un valor de offset previamente fijado (21):

$$violation = \frac{SO - SO_{max}}{SO_{max}} \qquad \textbf{(19)}$$

$$violation = \frac{TV - TV_{max}}{TV_{max}} \qquad \textbf{(20)}$$

$$offset = 10^6 \qquad SO_{max} = 0.3 \qquad TV_{max} = 100 \qquad \textbf{(21)}$$

$$F(x) = f(x) + offset \qquad \textbf{(22)}$$

$$F(x) = f(x) + violation \cdot offset \qquad \textbf{(23)}$$

Si solo se añade el “offset” (22) como penalización, se descartarían las que no cumpliesen las restricciones. Al añadir el “violation” (23) se obtiene una graduación del grado de incumplimiento de las restricciones.

A continuación, se procede a repetir los ensayos anteriores pero con las restricciones mencionadas (Tabla 6 y Tabla 7).

Tabla 6.- Resultados algoritmo genético con restricciones sobre la sobreoscilación y Variación Total (50 individuos, 50 iteraciones)

Prueba	**λ_d**	**λ_r**	**Función objetivo**	**Tiempo Cómputo**
Ensayo 1	0,5173	0,0000	44,8127	844,7813
Ensayo 2	0,5069	0,0000	44,5978	846,3438
Ensayo 3	0,5081	0,0000	44,6235	846,6563
Ensayo 4	1,2818	0,3116	49,6979	844,3594
Ensayo 5	0,5182	0,0095	44,8309	843,0156
Ensayo 6	1,2424	0,2667	49,1698	844,5625
Ensayo 7	1,2402	0,2676	49,1204	844,7344
Ensayo 8	0,5143	0,0512	44,7231	843,6406
Ensayo 9	0,5151	0,0465	44,7494	874,3594
Ensayo 10	1,2400	0,2666	49,1228	1011,7344
Min	***0,5173***	***0,0000***	**44,8127**	**844,7813**
Max	***1,2818***	***0,3116***	**49,6979**	**1011,7344**
Media			**46,5448**	**864,41875**

Tabla 7.- Resultados algoritmo genético con restricciones sobre la sobreoscilación y Variación total (10 individuos, 50 iteraciones)

Prueba	**λ_d**	**λ_r**	**Función objetivo**	**Tiempo Cómputo**
Ensayo 1	1,5563	0,6460	53,7532	170,7500
Ensayo 2	1,6752	0,7973	55,5488	169,9844
Ensayo 3	2,6508	2,1551	69,9092	169,5156
Ensayo 4	1,8068	0,9704	57,4959	182,7031
Ensayo 5	1,5582	0,6394	53,8227	170,0625
Ensayo 6	2,1021	1,3714	61,8176	168,8438
Ensayo 7	1,8129	0,9727	57,6112	169,2031
Ensayo 8	1,2702	0,2992	49,5434	171,8594
Ensayo 9	1,1099	0,0901	61,8906	169,9688
Ensayo 10	1,1200	0,1185	61,9274	168,5625
Min	***1,2702***	***0,2992***	**49,5434**	**168,5625**
Max	***2,6508***	***2,1551***	**69,9092**	**182,7031**
Media			**58,3320**	**171,1453**

Después de realizar varias simulaciones y encontrar un mínimo, se simulan pruebas con valores aleatorios para el modelo (Tabla 8):

K = rand()*2+1;

Tau = rand()*5+7;

T = rand()*2.5+1;

Tabla 8.- Valores para 100 iteraciones sin restricciones

	Máximo	**Mínimo**	**Media**
Ensayo 1	187.4398	47.7746	102.4646
Ensayo 2	182.0267	46.1573	99.4787
Ensayo 3	186.6143	44.7705	99.7931
Ensayo 4	180.3706	43.5894	98.1543
Ensayo 5	173.9032	45.8099	101.9850
Ensayo 6	186.3650	45.0674	104.5471
Ensayo 7	180.0710	47.4405	97.6711

Como se puede ver, todos los resultados obtenidos empeoran el valor óptimo F(x) del AG, que era el objetivo principal de estos ensayos.

3.4. Control predictivo SISO

Para empezar, se deben ajustar el tamaño de los horizontes de control. El horizonte de control c para sistemas de primer orden será dos, puesto que un horizonte mayor no implicará una mejora. Además, debido a la forma de implementar el control predictivo, se ha vuelto a emplear el simulink, a pesar de que es posible que se obtuviese una mejora en los tiempos si se eliminase y la implementación fuese mediante código exclusivamente.

A continuación se debe comprobar la respuesta del proceso a un escalón (véase Ilustración 22). En 82 segundos se alcanza el régimen estacionario, puesto que el periodo de muestreo es Ts = 2s, para asegurar que llega correctamente, se seleccionará que el horizonte p sea aproximadamente la mitad de dicho valor: 90/2 = 45.

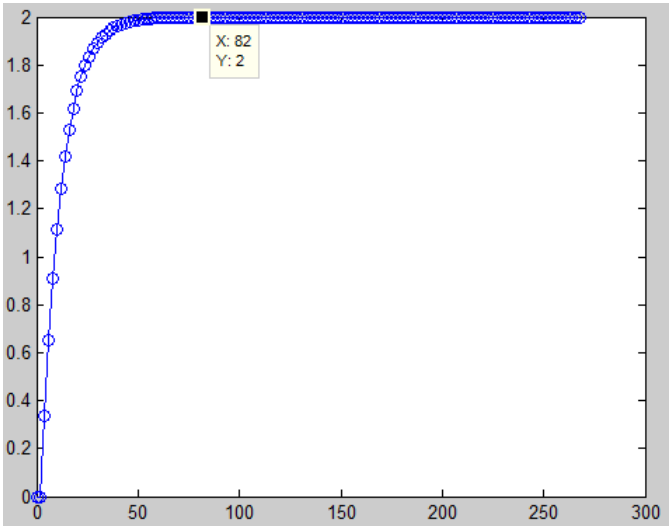

Ilustración 22.- Respuesta escalón del proceso.

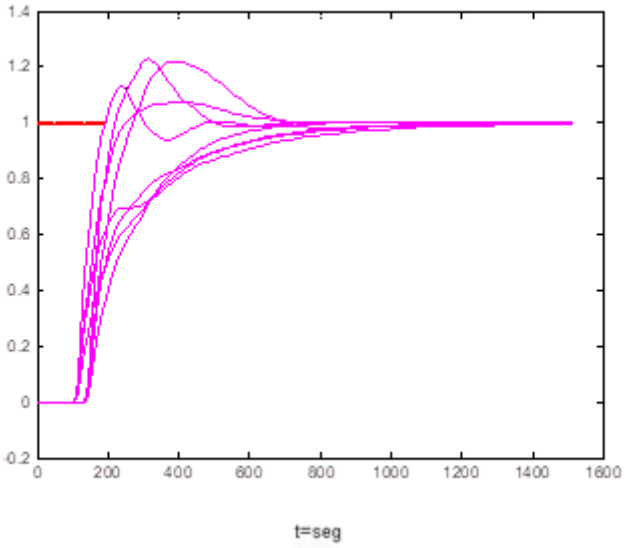

Ilustración 23.- Respuesta a prueba de sistema con control predictivo en un SISO

Ajustando de forma empírica el modelo nominal y los parámetros para que se consiga una mejor respuesta se alcanza un resultado similar al obtenido en el artículo en el que basamos estas pruebas [10]. Las comparativas se pueden observar en las siguientes figuras:

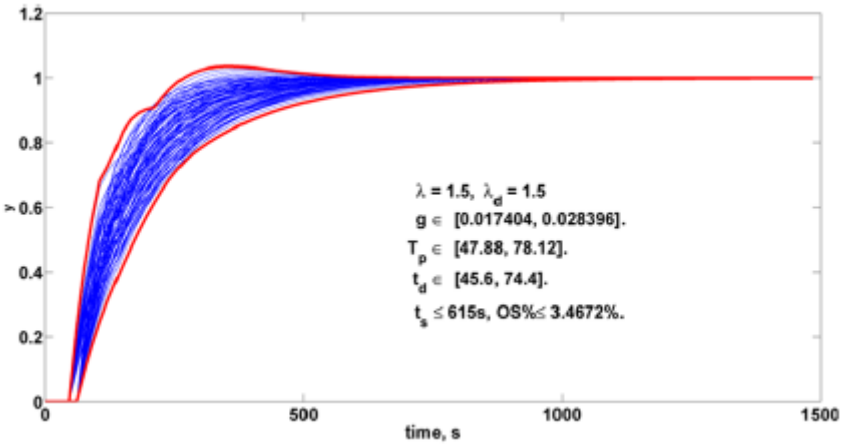

Ilustración 24.- Ejemplo artículo (a). [10]

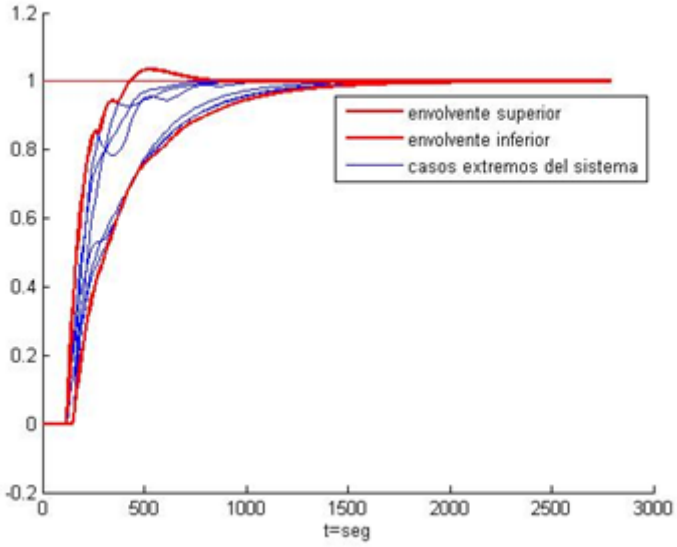

Ilustración 25.- Simulación SISO control predictivo (a)

La solución obtenida alcanza un tiempo de establecimiento menor que el marcado en el artículo pero a costa de un pequeño incremento en la sobreoscilación.

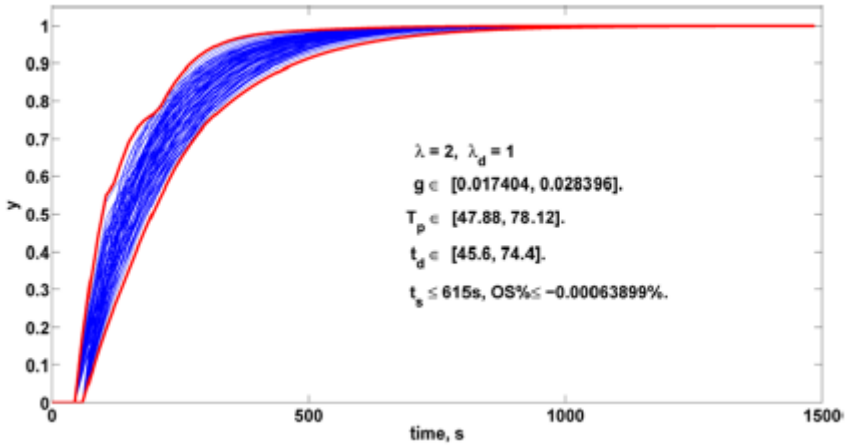

Ilustración 26.- Ejemplo artículo (b). [10]

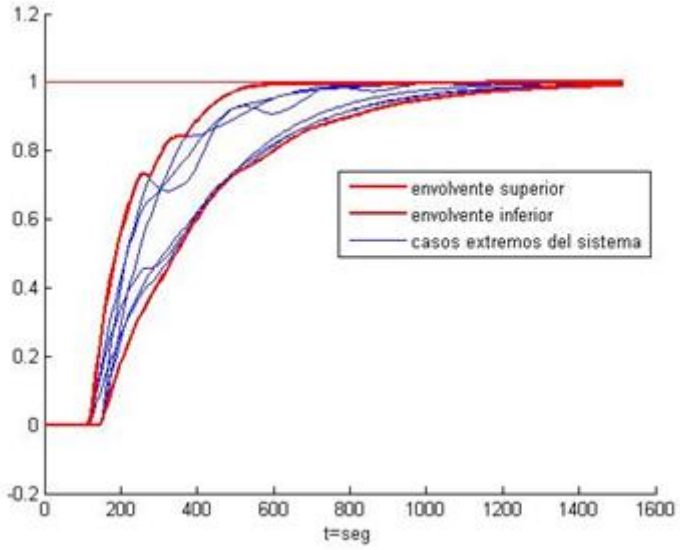

Ilustración 27.- Simulación SISO control predictivo (b)

La solución obtenida en este caso es similar si se compara tanto el tiempo de establecimiento como la sobreoscilación.

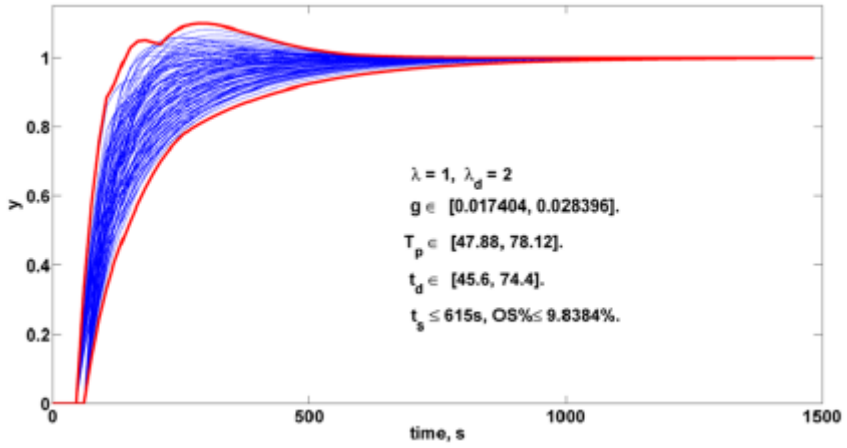

Ilustración 28.- Ejemplo artículo (c). [10]

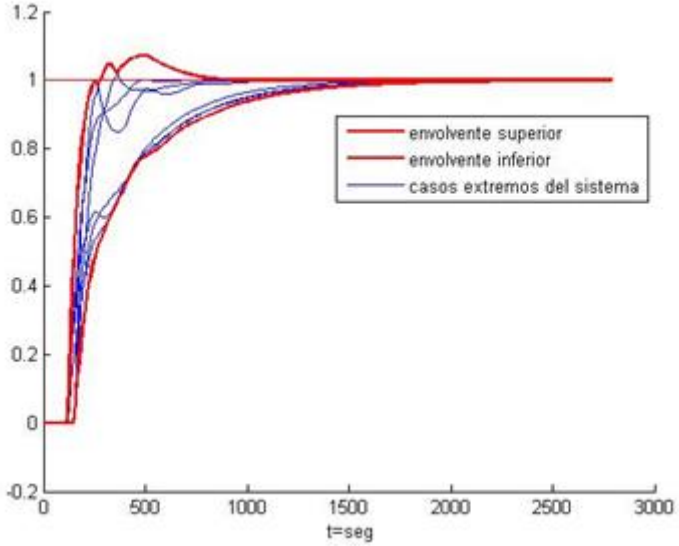

Ilustración 29.- Simulación SISO control predictivo (c)

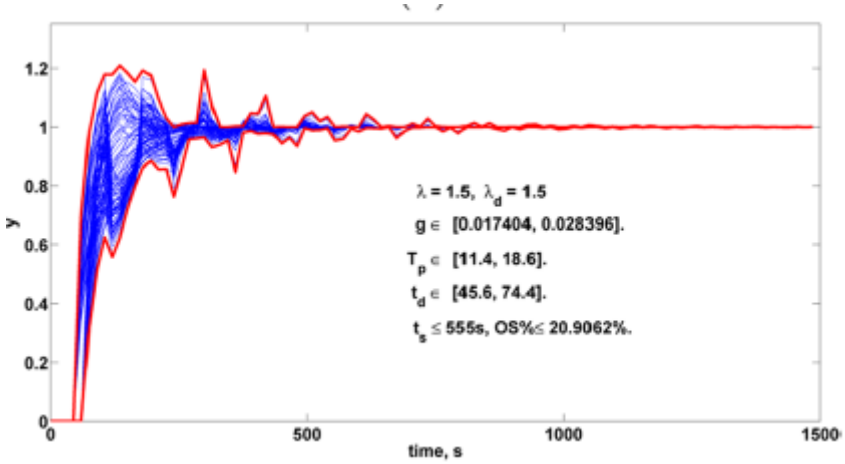

Ilustración 30.- Ejemplo artículo (d). [10]

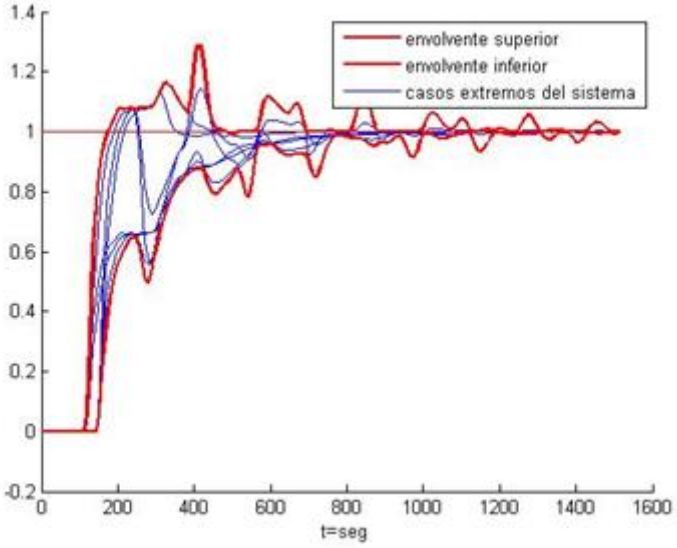

Ilustración 31.- Simulación SISO control predictivo (d)

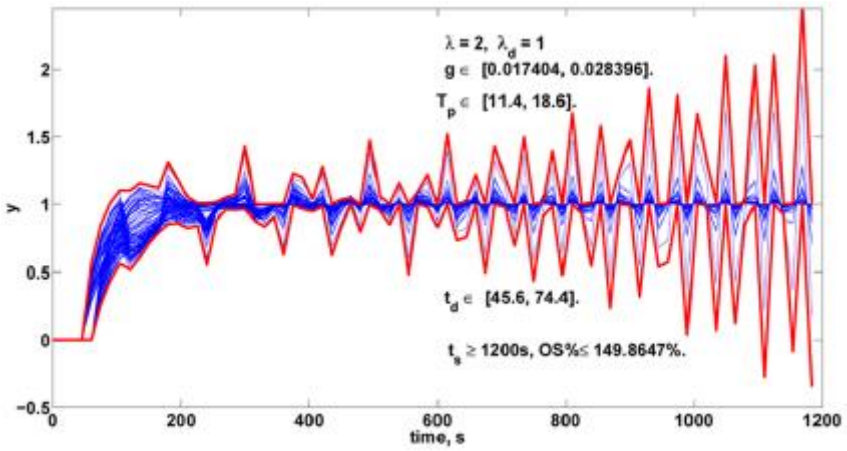

Ilustración 32.- Ejemplo artículo (e). [10]

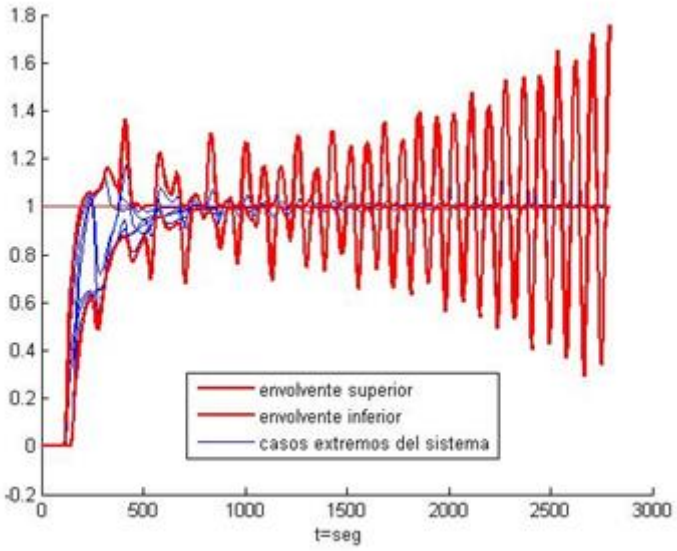

Ilustración 33.- Simulación SISO control predictivo (e)

Al comparar las respuestas se ve que se obtiene una aproximación similar, pues la sobreoscilación tiende a inestabilizarse a velocidad similar que en la solución aportada por el artículo.

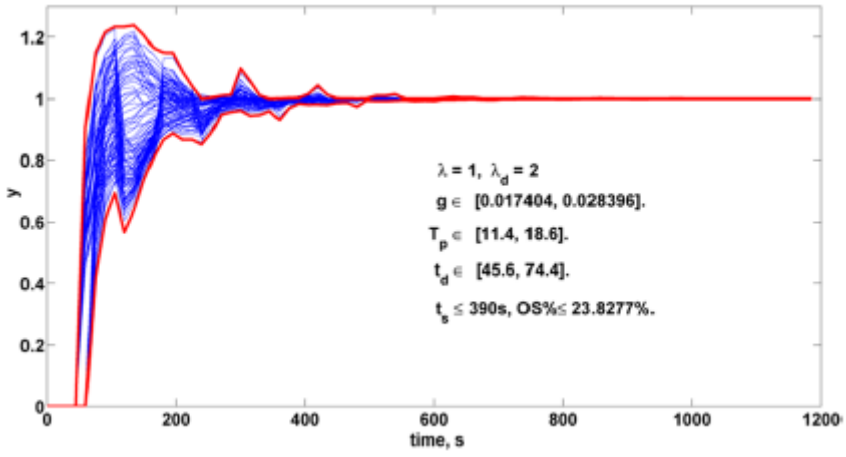

Ilustración 34.- Ejemplo artículo (f). [10]

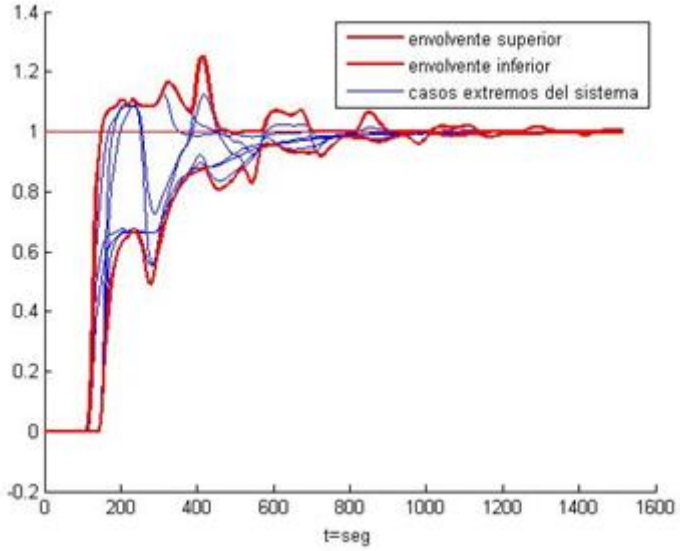

Ilustración 35.- Simulación SISO control predictivo (f)

Los resultados obtenidos en los casos siguientes son similares con algún pico de sobreoscilación puntual, por lo que el resultado obtenido cumple con las expectativas.

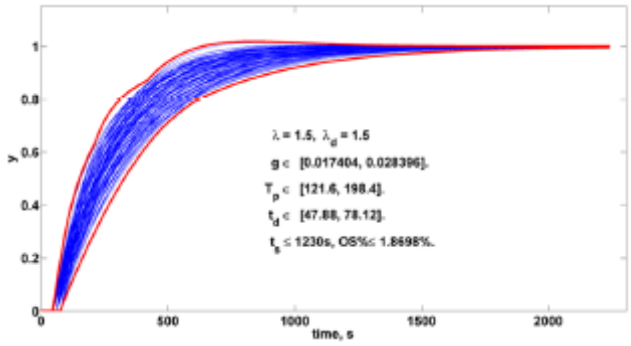

Ilustración 36.- Ejemplo artículo (g). [10]

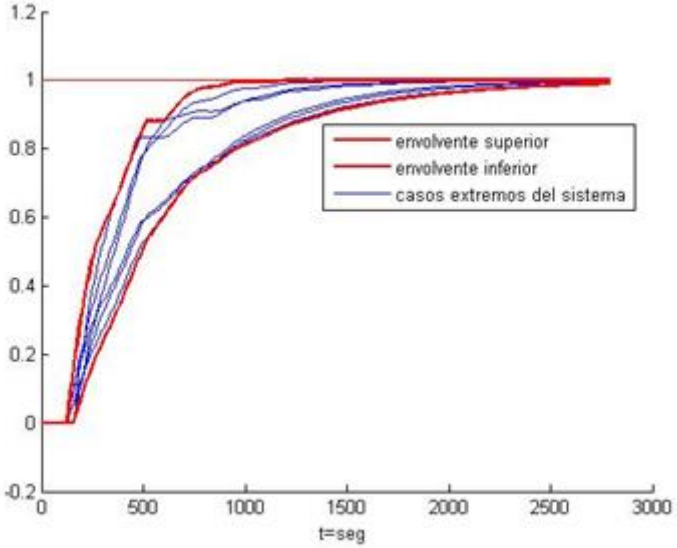

Ilustración 37.- Simulación SISO control predictivo (g)

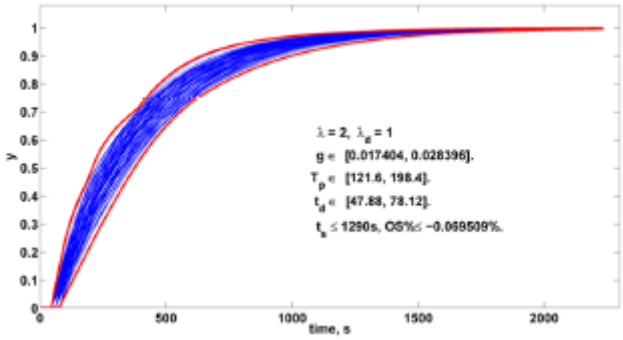

Ilustración 38.- Ejemplo artículo (h). [10]

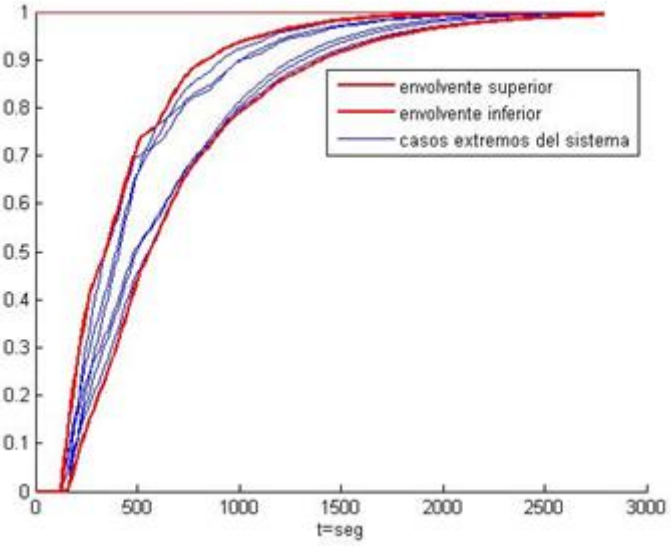

Ilustración 39.- Simulación SISO control predictivo (h)

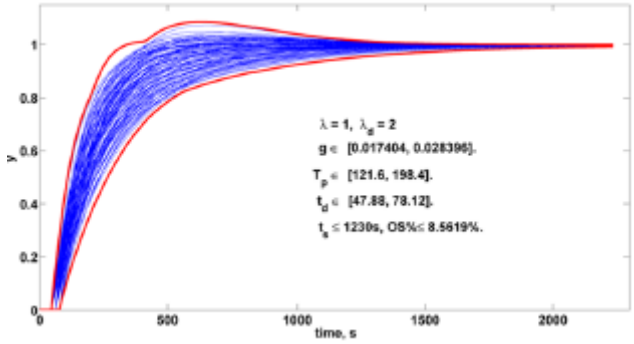

Ilustración 40.- Ejemplo artículo (i). [10]

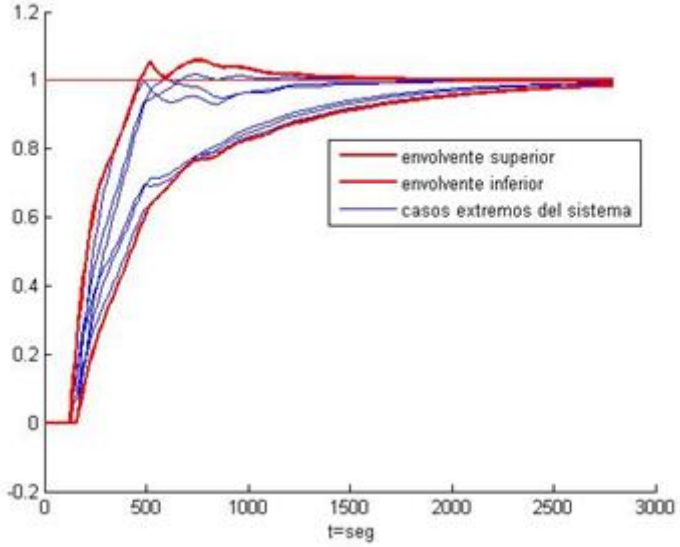

Ilustración 41.- Simulación SISO control predictivo (i)

3.5. Control predictivo MIMO

Al ejecutar los algoritmos desarrollados se implementa el control predictivo, ajustando de forma empírica sus correspondientes restricciones a las salidas, acciones de control, etc.

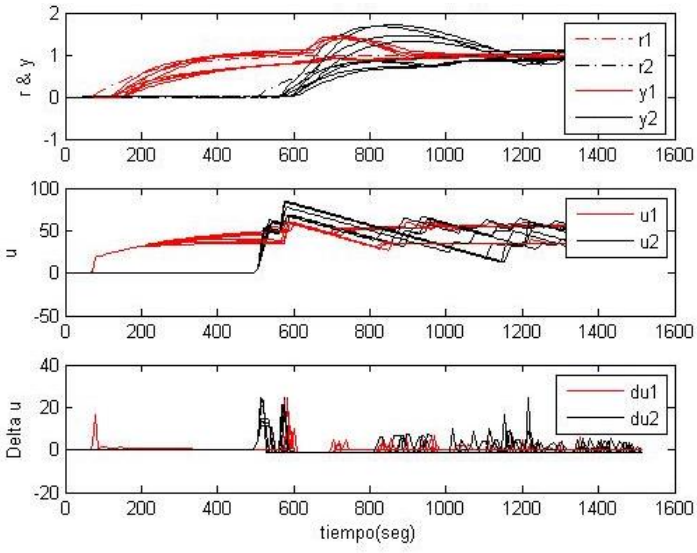

Ilustración 42.- Respuesta sistema MIMO + control predictivo

De la misma forma que en los casos previos, se puede ejecutar un algoritmo genético para la optimización del proceso sin restricciones (véase Tabla 9) o con ellas (véase Tabla 10):

Tabla 9.- Prueba MIMO sin restricciones en el AG

Prueba	**λ_d**	**λ_r**	**Función objetivo**	**Tiempo Cómputo**
Ensayo 1	1,0748	1,4363	1584,0000	198314,3906
Ensayo 2	1,1508	1,4508	1584,0000	199362,6094
Ensayo 3	1,0733	1,4363	1584,0000	199729,3125
Ensayo 4	2,2743	1,4447	1816,0000	199549,7500
Ensayo 5	1,0691	1,4366	1584,0000	209946,0625
Ensayo 6	1,0848	1,4352	1584,0000	219285,7031
Ensayo 7	1,0912	1,4372	1584,0000	219573,7031
Ensayo 8	1,1534	1,4507	1584,0000	219433,1563
Ensayo 9	1,0885	1,4379	1584,0000	219430,3906
Ensayo 10	1,0817	1,4365	1584,0000	219635,3594
Min	***1,0748***	***1,4363***	**1584,0000**	**198314,3906**
Max	***2,2743***	***1,4447***	**1816,0000**	**219635,3594**
Media			**1607,2000**	**210426,0438**

Tabla 10.- Prueba MIMO con restricciones en SO

Prueba	**λ_d**	**λ_r**	**Función objetivo**	**Tiempo Cómputo**
Ensayo 1	1,0780	1,4378	1584,0000	203784,0781
Ensayo 2	1,0635	1,4355	1584,0000	203246,3125
Ensayo 3	1,1539	1,4504	1584,0000	202557,6406
Ensayo 4	1,0634	1,4359	1584,0000	203012,2969
Ensayo 5	1,1574	1,4506	1584,0000	202724,3281
Ensayo 6	1,1663	1,4513	1584,0000	202900,7813
Ensayo 7	1,0793	1,4368	1584,0000	202942,3125
Min	***1,1539***	***1,4504***	**1584,0000**	**202557,6406**
Max	***1,0780***	***1,4378***	**1584,0000**	**203784,0781**
Media			**1584,0000**	**203023,9643**

A modo de prueba final del MIMO, se ha diseñado un sistema de dos entradas y dos salidas con el que poder ver los efectos del mismo (24).

$$G = \begin{bmatrix} \frac{K_1}{\tau_1 s+1} e^{-T_1 s} & 0 \\ 0 & \frac{K_2}{\tau_2 s+1} e^{-T_2 s} \end{bmatrix} \quad \textbf{(24)}$$

Este sistema estará definido por los peores casos posibles para los que se simulará (25)(26)(27)(28)(29)(30):

$$K_1 \in [0.017404\ 0.028396] \quad (25)$$
$$\tau_1 \in [47.88\ 78.12] \quad (26)$$
$$T_1 \in [45.6\ 74.4] \quad (27)$$

$$K_2 \in [0.017404\ 0.028396] \quad (28)$$
$$\tau_2 \in [121.6\ 198.4] \quad (29)$$
$$T_2 \in [47.8\ 78.12] \quad (30)$$

Si se simula este Sistema se obtendrá como envolvente común una similar a la mostrada en la Ilustración 43

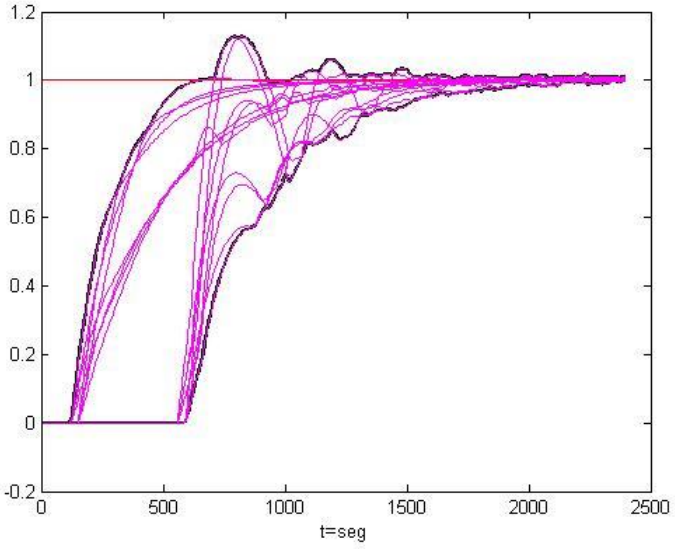

Ilustración 43.- Envolvente común MIMO 2x2

Para que la misma sea más claramente observable, se puede dividir en dos, una por cada una de las salidas (véanse Ilustración 44 e Ilustración 45)

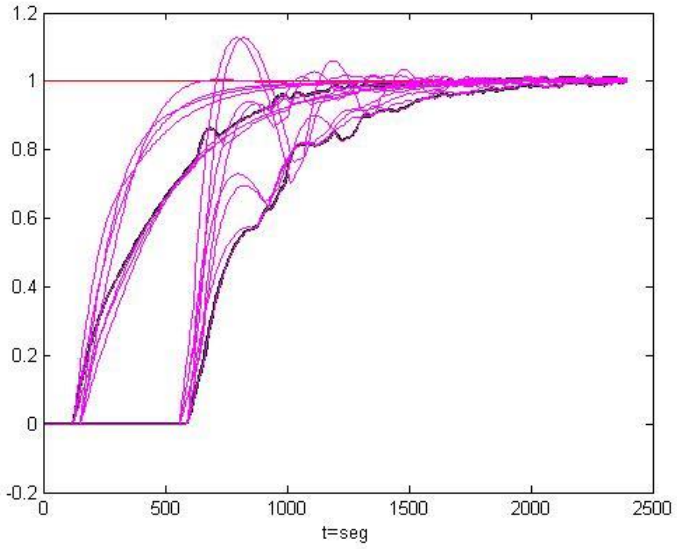

Ilustración 44.- Envolvente MIMO 2x2 G11

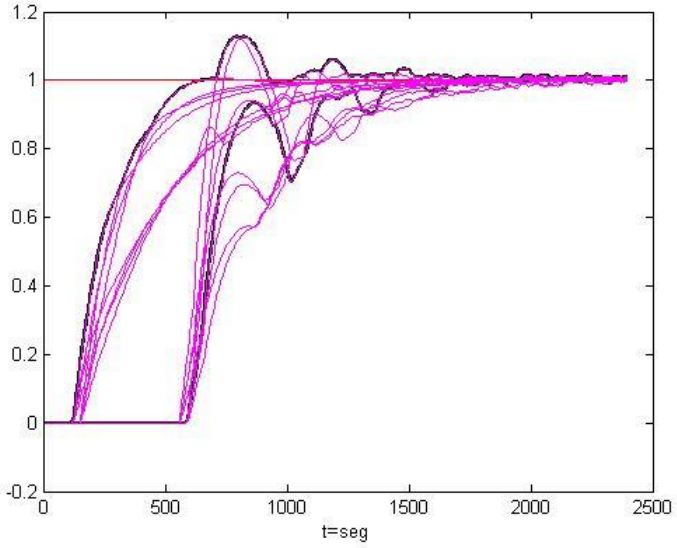

Ilustración 45.- Envolvente MIMO 2x2 G22

ESTA PÁGINA HA SIDO INTENCIONALMENTE DEJADA EN BLANCO

4. Conclusiones

En este trabajo se ha considerado un problema de optimización basado en minimizar los peores casos para el tiempo de establecimiento, la sobreoscilación y la variación total con la ayuda de algoritmos genéticos.

En un primer momento, se ha simulado probando el control tradicional PID en un sistema SISO para comprobar que los filtrados en los que se basa la sintonización de la estructura de control son eficaces y posteriormente se ha probado a incorporar un control predictivo del tipo DMC y QDMC tanto en SISO como en MIMO.

Se ha probado que la implementación en simulink ralentiza los tiempos de cómputo por lo que es recomendable en la medida de lo posible evitarlo. A pesar de esto, para los casos de control predictivo ha sido necesario por la forma de implementar el algoritmo de control predictivo.

Mediante los algoritmos se alcanzan soluciones óptimas para los problemas planteados y posteriormente se verifica a modo de comprobación que dichas soluciones cumplen los requisitos planteados por el algoritmo.

Si se repasan los resultados alcanzados a lo largo de todo el trabajo se puede observar que las respuestas obtenidas son más suaves y más intuitivamente controlables (sobre todo mediante el tema de restricciones tanto en el algoritmo como en los parámetros del control predictivo).

En la comparativa para el caso SISO, se obtienen resultados muy similares a los obtenidos en los artículos de referencia, demostrando la efectividad de este control para este tipo de sistemas.

En el caso MIMO se ha implementado y probado el algoritmo para un caso inicial de dos entradas y dos salidas, obteniendo su función de coste y sus correspondientes envolventes.

Como posibles trabajos futuros, podría suprimirse el simulink de la parte de control predictivo. Además, si se ampliase al caso tres entradas y dos salidas, se podría hacer una comparativa con el artículo [1].

5. Bibliografía

[1] He, N., Shi, D., Forbes, M., Backström, J. and Chen, T. (2016). Robust tuning for machine-directional predictive control of MIMO paper-making processes. *Control Engineering Practice*, 55, pp.1-12.

[2] Fraser, A. S., "Simulation of genetic systems by automatic digital computers. I. Introduction," Aust. J. Biol. Sci., vol. 10, pp. 484–491, 1957.

[3] J. H. Holland, Adaptation in Natural and Artificial Systems, 1962.

[4] G. d. C. P. y. C. Heurístico, «CPOH Control y Optimización,» [En línea]. Disponible en: http://cpoh.upv.es/. [Último acceso: 6 Junio 2017].

[5] F. Xavier Blasco Ferragud. PhD Tesis 1999 (in Spanish). Editorial UPV. ISBN 84-699-5429-6, Control predictivo basado en modelos mediante técnicas de optimización heurística. Aplicación a procesos no lineales y multivariables. Disponible en: http://hdl.handle.net/10251/15995. [Último acceso: 6 Junio 2017].

[6] Deb, K. (2000). An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, 186(2-4), pp.311-338.

[7] Chu, D., Forbes, M. and U. Backstrom, J. (2017). *Technique for converting a model predictive control (MPC) system into an explicit two-degrees of freedom (DOF) control system*. 9557724B2.

[8] Es.mathworks.com. (2017). *MathWorks - Makers of MATLAB and Simulink*. [En línea] Disponible en: https://es.mathworks.com [Último acceso: 12 Jul. 2017].

[9] He, N., Shi, D., Wang, J., Forbes, M., Backström, J. and Chen, T. (2015). Automated Two-Degree-of-Freedom Model Predictive Control Tuning. Industrial & Engineering Chemistry Research, 54(43), pp.10811-10824.

[10] Shi, D., Wang, J., Forbes, M., Backström, J. and Chen, T. (2015). Robust Tuning of Machine Directional Predictive Control of Paper Machines. Industrial & Engineering Chemistry Research, 54(15), pp.3904-3918.

6. Anexos

A. Código ejecución del Algoritmo Genético

```
%% Parámetros Algoritmo Genético Departamento (BasicGA)
gaDat.Objfun='TsgaSISO';                % Función objetivo SISO
% gaDat.Objfun='FobjMIMOR';             % Función objetivo MIMO
lb=[0.1 0.1];                             % Límites
ub=[3 3];
gaDat.FieldD=[lb; ub];
gaDat.NIND=50;                          % N° individuos
% gaDat.NIND=10;
gaDat=ga(gaDat);                        % Ejecución del GA
```

B. Código función objetivo de control PID en SISO con simulink

```
function [TS] = Tsmega(lambdas)

% Parámetros referencias
global Ts
global r
global Fref
global t
global Tstop
global steptimeDV
global valueDV
global steptimeUDV
global valueUDV
global outputnoisegain
global inputnoisegain

% Parámetros modelo
global K
global T
global Tau
global s
global PID
global PIDd

global guss
% global gu
% global gu2
% global gdv
% global gudv
% global filtergu
% global filterdgu

% PID
% global Kc
%PID ajustado mediante toolbox pid matlab
% global P
% global I
% global D
```

```
% global N
% global Ti
% global Td
% global Kpre
%sin prealimentación

global datalambdas
global TVdata
global SOdata
global Testdata
global Y
global yRP

% sim('simtfmpid')
%dibuja(Simul,'k','AÑADIR DATOS')
%%

lambdaDis=lambdas(1);
lambdaRef=lambdas(2);


%Inicializo parametros
Testdata = [];  TestWCmin = 1000000;  TestWCmax = -10000000;
% Tiempo de establecimiento
SOmin = 0;    SOmax = 0;   SOexist = 0;
% Sobreoscilación
TVdata = [];    TVWCmin = 1000000;   TVWCmax = -1000000;
% Variación Total
yRP = r(length(r));
% Valor en regimen permanente
indK=1; indTau=1; indT=1;   indTVsol=1;
% Indicadores para los bucles de los parametros
ymax = 0; ymin = 0;
% Valores max y min de salida para envolvente
indicador = 0;
m = 0;
mmax= length(lambdaRef) * length(K) * length(Tau) * length(T);
% N° de funciones
k = 1;
%kmax = length(y.time);     % o length(y.data)
% N° de pasos o instantes muestreados

indK = 1;
n=1;    n1=1;

while indK < length(K) + 1
    indTau = 1;

    datalambdas(indTau,:) = lambdas;

    while indTau < length(Tau) + 1
        indT = 1;

        while indT < length(T) + 1
            indicador = indicador + 1;
            % Actualizo modelos con los datos de cada caso
            gu = K(indK)/((Tau(indTau)*s+1));
% fdt y/u proceso real
```

```
            gu.OutputDelay = T(indT);
% añadimos retardo
            gdv = tf(1,[1 1]);
% fdt y/dv (sin perturbación medible)
            gudv = tf(1,[1 1]);
% fdt y/udv (sin perturbación no medible)
            gu2 = K(indK)/((Tau(indTau)*s+1));
% nominal
            gu2.OutputDelay = T(indT);
% añadimos retardo
            gume(indicador,1) = gu;


            %Definimos el modelo de los filtros
            %Reference Tracking Filter
            filtergu=1/(((Tau(indTau)^lambdaRef)*s+1));% fdt y/u
            %Disturbancce Rejection Filter
            filterdgu=1/(((Tau(indTau)^lambdaDis)*s+1));% fdt y/u
            dfiltergu=c2d(filtergu,Ts);
            dfilterdgu=c2d(filterdgu,Ts);

            Freferencia = lsim(dfilterdgu,r,t);
            gudis = c2d(gu,Ts);
            gu2dis = c2d(gu2,Ts);
            gu1C = PID*gu;
            gu1Cd = c2d(gu1C,Ts);
            gu2C = PID*gu2;
            gu2Cd = c2d(gu2C,Ts);
            guC = gu1Cd - gu2Cd;
            proc1 = feedback(gu1Cd,dfiltergu,-1);
            [param, t] = lsim(proc1,Freferencia,t);
            PARAM = lsiminfo(param,t);
            TESTAB = PARAM.SettlingTime;

            %figure(indT)   % Para plotear en cada ventana
            %             hold on
            %             lsim(proc1,Freferencia,t)

            %% Matriz G
            m = m+1;
            G(m,:) = gu;    %+gdv+gudv;

            %% lsiminfo
%             lsim(proc1,Freferencia,t)
            [param, t] = lsim(proc1,Freferencia,t);
            Y(m,:) = param;
            PARAM = lsiminfo(param,t);
            TESTAB = PARAM.SettlingTime;
            %             SOSC = Y(PARAM.Max) - yRP;
            %kmax = PARAM.MaxTime;
            kmax = length(Freferencia);
            %% Sobreoscilación peores casos matricialmente
            %SOmaxm = max(y.data)
            posSO = find(Y(m,:) > yRP);
            if isempty(posSO)
                SOdata(m,1) = 0;
                SOdata(m,2) = 0;
            end
            if posSO ~= 0
                SOdata(m,1) = max(Y(m,posSO:kmax) - yRP);
```

```
                SOdata(m,2) = min(Y(m,posSO:kmax) - yRP);
            end



            %% Tiempo Establecimiento base de datos
            posRPprev = find(Y(m,:) < 0.98*yRP | Y(m,:) > 1.02*yRP);
% Posición en la que se supera el % de la franja

            %posTest = find(Y(m,:) > 0.98*yRP & Y(m,:) < 1.02*yRP);
            if (posRPprev(end) == size(Y(m,:),2))
                posRP(m,:) = posRPprev(end);
            else
                posRP(m,:) = posRPprev(end)+1;
            end
            %Testdata(m,:) = t(posRP(m));
            Testdata(m,:) = TESTAB;

            % Constante de tiempo (0.1 es el intervalo transcurrido
entre dos medidas,
            % se le resta 1, porque los indices empiezan en 1):
            % tauEstim = 0.1*(posTest(1) - 1); % o t(posTest(1))

            %% Variación Total base de datos

            TVdata(m,:) = sum(Y(m,1:posRP(m)))-
sum(Y(m,1:posRPprev(end)));

            indT = indT + 1;
        end
        indTau = indTau + 1;
    end
    indK = indK + 1;
end

guss = ss(gume);

TS = max(Testdata);
end
```

C. Código del algoritmo de control PID de un SISO con Simulink

```
%% Definimos variables globales para poder emplearlas en la función

clear all
close all

% cputsfunga1 = cputime;        % tiempo de cpu invertido en la
ejecucion del mencionado conjunto de operaciones

global datalambdas

% Parámetros referencias
```

```
global Ts
global r
global Fref
global t
global Tstop
global steptimeDV
global valueDV
global steptimeUDV
global valueUDV
global outputnoisegain
global inputnoisegain

% Parámetros modelo
global K
global T
global Tau
global s
global guss
% global gu
% global gu2
global PID
global PIDd
% global gdv
% global gudv
% global filtergu
% global filterdgu

%% PID
% global Kc
%PID ajustado mediante herramienta de ajuste de pid matlab
% global P
% global I
% global D
% global N
% global Ti
% global Td
% global Kpre
% sin prealimentación

%% Base de datos de SO,Test,TV
global TVdata
global SOdata
global Testdata
global Y
global yRP
global kmax

%definición inicial del modelo del proceso
s=tf('s');
% Tau=10;
% K=2;
% gu=K/((Tau*s+1));% fdt y/u
% gu.OutputDelay=3;% añadimos retardo
% gdv=tf(1,[1 1]);% fdt y/dv (sin perturbación medible)
% gudv=tf(1,[1 1]);% fdt y/udv (sin perturbación no medible)

%definimos referencia y perturbaciones
Ts=1;
%inicializar;
```

```
r=[zeros(3/Ts,1); 1*ones(450/Ts,1)];
% referencia1
%r1=[zeros(2/Ts,1); 12*ones(38/Ts,1); zeros(12/Ts,1);
12*ones(48/Ts,1)];     % referencia2
t=(0:Ts:Ts*size(r,1)-Ts)';
Tstop=t(end);
steptimeDV=0;valueDV=5*rand;
%sin perturbaciones dv y udv
steptimeUDV=0;valueUDV=-5*rand;
outputnoisegain=0;
%el rudio está entre +-outputnoisegain
inputnoisegain=0;
%el rudio está entre +-inputnoisegain

% %Definimos posibles modelos de los filtros
% %Reference Tracking Filter
% %lambdaRef=1.25;
% % lambdaRef=0.7;
% filtergu=1/(((Tau^lambdaRef)*s+1));% fdt y/u
%
% %Disturbancce Rejection Filter
% %lambdaDis=0.25;
% % lambdaDis=0.8;
% filterdgu=1/(((Tau^lambdaDis)*s+1));% fdt y/u

%simulamos y representamos resultados
%figure

%% PID ajustado mediante toolbox pid matlab
Kc=0.69511;
Ki=0.057993;
Kd=0.33328;
P=Kc;
I=Ki;
D=Kd;
PID = pid(Kc,Ki,Kd,t(end));
PIDd = c2d(PID,Ts,'zoh');

N=0.168028164538568;
Kpre=0;%sin prealimentación
%sim('simtfmpid')
%dibuja(Simul,'k','AÑADIR DATOS')

%% Inicializo parametros
% Testdata = [];  TestWCmin = 1000000;  TestWCmax = -10000000;
% Tiempo de establecimiento
% SOmin = 0;    SOmax = 0;   SOexist = 0;
% Sobreoscilación
% TVdata = [];    TVWCmin = 1000000;   TVWCmax = -1000000;
% Variación Total
% yRP = r(length(r));
% Valor en regimen permanente
% indK=1; indTau=1; indT=1;   indTVsol=1;
% Indicadores para los bucles de los parametros
% ymax = 0; ymin = 0;
% Valores max y min de salida para envolvente

%% Valores extremos de los parametros
% K = 1:1:3;
% Tau = 7:1:12;
```

```
% T = 1:0.5:3.5;
K = [1 3];
Tau = [7 12];
T = [1 3.5];

% m = 0;
% mmax= length(lambdaRef) * length(K) * length(Tau) * length(T);
% N° de funciones
% k = 1;
% kmax = length(y.time);     % o length(y.data)
% N° de pasos o instantes muestreados

% lambdaRef=0.7;
% lambdaDis=0.8;

% lambdaRef=0;
% lambdaDis=0;
% lambdas = [lambdaRef lambdaDis];
%
% lambdas = [0.246383536512386 0.512123142175763];
% lambdas = [0.277651283790912 0.51438810631008];
%
% TS = Tsga(lambdas)
%
% %% Peor caso de Total Variation
% TVWCmin = min(TVdata);
% TVWCmax = max(TVdata);
% TVWC(1,:) = [TVWCmax TVWCmin]
%
% %% Peor caso Tiempo de Establecimiento
% TestWCmax = max(Testdata);
% TestWCmin = min(Testdata);
% TestWC(1,:) = [TestWCmax TestWCmin]
%
% %% Peor caso Sobreoscilacion
% SOWCmax = max(SOdata(:,1));
% SOWCmin = min(SOdata(:,2));
% SOWC(1,:) = [SOWCmax SOWCmin]
%
%% Extremos de la envolvente
% Ymax = max(Y);
% Ymin = min(Y);
%
% %% Ploteamos la envolvente
% figure
% plot(Ymax,'b-')
% hold on
% plot(Ymin,'b-')
% plot([1:1:length(Y)],yRP,'r--')
% % % plot(Ynom.data, 'g-')

%% Algoritmo Genético Matlab común
% [x,fval,exitflag,output,population,scores] =
ga(fitnessfcn,nvars,...)
% InitialPopulationMatrix = 2;
% A = [1 0; -1 0; 0 1; 0 -1];
% b = [1; 0; 1; 0];
% [x,fval,exitflag,output,population,scores] = ga(@Tsfunga, 2, A,b)
% lb = [];
% ub = [];
```

```
%% Basic GA parameters
% gaDat.Objfun='Tsfunga';
% gaDat.Objfun='Tsga';
gaDat.Objfun='Tsmega';
lb=[0 0];
ub=[3 3];
gaDat.FieldD=[lb; ub];
% gaDat.NIND=50;
gaDat.NIND=10;
% Execute GA
gaDat=ga(gaDat);

% %% Tiempo de coste computacional
% cputsfunga = cputime-cputsfunga1
```

D. Código función objetivo de control PID en SISO sin simulink

```
function [J] = FobjRSOTV(lambdas)

% Parámetros referencias
global Ts
% Tiempo muestreo
global r
% Referencia
global Fref
% Referencia Filtrada
global t
% Tiempo simulación
global Tstop
% Tiempo parada

% Modelo nominal
global K0
global Tau0
global T0

% Parámetros modelo
global K
global T
global Tau
global s
global PID
global PIDd

global G0
global G0dis

global TPdata
global datalambdas
global TVdata
global SOdata
global Testdata
global Y
global yRP
```

```
global k

%% Definición de lambdas
lambdaDis=lambdas(1);
lambdaRef=lambdas(2);

%% Indicadores para los bucles de los parametros
indK=1; indTau=1; indT=1;
m = 0;
mmax= length(lambdaRef) * length(K) * length(Tau) * length(T);
% N° de funciones
k = 1;
%kmax = length(y.time);     % o length(y.data)
% N° de pasos o instantes muestreados
kmax = length(r);

datalambdas(k,:) = lambdas;
k = k+1;

%Definimos el modelo de los filtros
%Reference Tracking Filter
filtergu=1/(((Tau0*lambdaRef)*s+1));% fdt y/u
filtergu.OutputDelay = T0;
%Disturbancce Rejection Filter
filterdgu=1/(((Tau0*lambdaDis)*s+1));% fdt y/u
filterdgu.OutputDelay = T0;
dfiltergu=c2d(filtergu,Ts);
dfilterdgu=c2d(filterdgu,Ts);

while indK < length(K) + 1
    indTau = 1;

    while indTau < length(Tau) + 1
        indT = 1;

        while indT < length(T) + 1
            % Actualizo modelos con los datos de cada caso
            G = K(indK)/((Tau(indTau)*s+1));
% Modelo
            G.OutputDelay = T(indT);
% Retardo

             Freferencia = lsim(dfilterdgu,r,t);
             Gdis = c2d(G,Ts);
% %                          CG0 = PID*G0;
% %                          CG = PID*G;
% %                          CGd = c2d(CG,Ts);
%                         error = CGd - CG0d;
%                         ef = error*dfiltergu;
%                         proc1 = feedback(CGd,CG0d+ef,-1);
%                         TP0 = proc1;

%% Continuo
%             Freferencia = lsim(filterdgu,r,t);
%             C = feedback(PID, G0-(filtergu*G0),-1);
%             TotC = feedback(C*G,filtergu);
%             TP = filterdgu*TotC;

%% Discreto
```

```
            C = feedback(PIDd, G0dis-(dfiltergu*G0dis),-1);
            TotC = feedback(C*Gdis,dfiltergu);
            TP = dfilterdgu*TotC;

            %step(TP)
            m = m+1;
             stepdata = stepinfo(TP);
             TSstepdata(m,:) = stepdata.SettlingTime;
             OSstepdata(m,:) = stepdata.Overshoot;

            %% figure(indT)   % Para plotear en cada ventana
            %             hold on
            %             lsim(proc1,Freferencia,t)

            %% Matriz G
            Gdata(m,:) = G0;     %+gdv+gudv;

            %% lsiminfo
%             lsim(TP,r,t,'y-')
%              hold on
%              lsim(TotC,Freferencia,t,'b--')
            [param, t] = lsim(TotC,Freferencia,t);
%             [param, t] = step(TP);
            Y(:,m) = param;
            PARAM = lsiminfo(param,t);
            %SOSC = Y(PARAM.Max) - yRP;

            %% Sobreoscilación peores casos matricialmente
            %SOmaxm = max(y.data)
            posSO = find(Y(:,m) > yRP);
            if isempty(posSO)
                SOdata(m,1) = 0;
                SOdata(m,2) = 0;
            end
            if posSO ~= 0
                SOdata(m,1) = max(Y(posSO:kmax,m) - yRP);
                SOdata(m,2) = max(yRP - Y(posSO:kmax,m));
            end

            %% Cálculo de posición Regimen Permanente
            posRPprev = find(Y(:,m) < 0.98*yRP | Y(:,m) > 1.02*yRP);
% Posición en la que se supera el % de la franja

            %% Tiempo Establecimiento base de datos
            %posTest = find(Y(m,:) > 0.98*yRP & Y(m,:) < 1.02*yRP);
            if (posRPprev(end) == size(Y,1))
                posRP(m,:) = posRPprev(end);
            else
                posRP(m,:) = posRPprev(end)+1;
            end
            %Testdata(m,:) = t(posRP(m));

            TESTAB = PARAM.SettlingTime;
            Testdata(m,:) = TESTAB;

            % Constante de tiempo (0.1 es el intervalo transcurrido
entre dos medidas,
            % se le resta 1, porque los indices empiezan en 1):
            % tauEstim = 0.1*(posTest(1) - 1); % o t(posTest(1))
```

```
            %% Variación Total base de datos

            TVdata(m,:) = sum(Y(1:posRP(m),m))-
sum(Y(1:posRPprev(end),m));

            indT = indT + 1;
        end
        indTau = indTau + 1;
    end
    indK = indK + 1;
end

SettlingTime = 'SettlingTime';
Overshoot = 'Overshoot';
TotalVariation = 'TotalVariation';
TPdata =
struct(SettlingTime,TSstepdata,Overshoot,OSstepdata,TotalVariation,TVd
ata);

%% Peor caso Sobreoscilacion
SOWCsup = max(SOdata(:,1));
SOWCinf = max(SOdata(:,2));
SOWC = max(SOdata(:,:));

%% Solución J para TS
TS = min(Testdata);
J = TS;

%% Parámetros
violation = 0;
offset = 1e6;
SO = max(SOWC);
TV = max(TVdata);

% con violation SO y TV
if SO > 0.3
    violation = (SO - 0.3)/0.3;
    J = J+violation*offset;
end
if TV > 1
    violation = (TV - 1)/1;
    J = J+violation*offset;
end

%% con offset
% if SO > 0.3
%     J = J+offset
% end
% if TV > 100
%     J = J+offset
% end

end
```

E. Código del algoritmo de control PID de un SISO sin Simulink

```
%% Definimos variables globales para poder emplearlas en la función

clear all
w = warning('query','last');id = w.identifier;warning('off',id);
%w.identifier = Control:ltiobject:UseSSforInternalDelay;
base = 1;
while base < 2
    % tiempo de cpu invertido en la ejecucion del mencionado conjunto
de operaciones
    cputsfunga1 = cputime;        % tiempo de cpu invertido en la
ejecucion del mencionado conjunto de operaciones

% Parámetros referencias
global Ts
global r
global Fref
global t
global Tstop
% global steptimeDV
% global valueDV
% global steptimeUDV
% global valueUDV
% global outputnoisegain
% global inputnoisegain

% Modelo nominal
global K0
global Tau0
global T0

% Parámetros modelo
global K
global T
global Tau
global s

global PID
global PIDd

global G0
global G0dis

% Base de datos de SO,Test,TV
global TPdata
global TVdata
global SOdata
global Testdata
global datalambdas
global Y
global yRP
global kmax

%definición inicial de función de transferencia
s=tf('s');

%% Referencia
Ts=2;
r=[zeros(20/Ts,1); 1*ones(250/Ts,1)];
```

```
% Ts=1;
% Tiempo de muestreo
% %inicializar;
% r=[zeros(3/Ts,1); 1*ones(450/Ts,1)];
t=(0:Ts:Ts*size(r,1)-Ts)';
Tstop=t(end);

%% Perturbaciones
% steptimeDV=0;valueDV=5*rand;
%sin perturbaciones dv y udv
% steptimeUDV=0;valueUDV=-5*rand;
% outputnoisegain=0;
%el rudio está entre +-outputnoisegain
% inputnoisegain=0;
%el rudio está entre +-inputnoisegain

%% Modelo extremos
K = [1 3];
Tau = [7 12];
T = [1 3.5];

%% Modelo Nominal
% K0, Tau0, T0 son los valores nominales del proceso
K0 = mean(K);
Tau0 = mean(Tau);
T0 = mean(T);
G0 = K0/((Tau0*s+1));
% fdt y/u proceso nominal
G0.OutputDelay = T0;
% Retardo
    G0dis = c2d(G0,Ts);


%% Modelo de los filtros
% %Reference Tracking Filter
% filtergu=1/(((Tau^lambdaRef)*s+1));% fdt y/u
%
% %Disturbancce Rejection Filter
% filterdgu=1/(((Tau^lambdaDis)*s+1));% fdt y/u

%simulamos y representamos resultados
%figure

%% PID ajustado mediante toolbox pid matlab
Kc=0.69511;
Ki=0.057993;
Kd=0.33328;
N=0.168028164538568;
Tf=1/N;
P=Kc;
I=Ki;
D=Kd;
% PID = pid(Kc,Ki,Kd,t(end));
PID = pid(Kc,Ki,Kd,Tf);
PIDd = c2d(PID,Ts,'zoh');
% PIDd = pid(Kc,Ki,Kd,Tf,Ts,'iFormula','ForwardEuler');

Kpre=0;
%prealimentación
```

```
%% Inicializo parametros
Testdata = [];
% Tiempo de establecimiento
SOdata = [];
% Sobreoscilación
TVdata = [];
% Variación Total
yRP = r(length(r));
% Valor en regimen permanente

% Testdata = [];  TestWCmin = 1000000;  TestWCmax = -10000000;
% Tiempo de establecimiento
% SOmin = 0;    SOmax = 0;   SOexist = 0;
% Sobreoscilación
% TVdata = [];    TVWCmin = 1000000;   TVWCmax = -1000000;
% Variación Total
% yRP = r(length(r));
% Valor en regimen permanente
% indK=1; indTau=1; indT=1;   indTVsol=1;
% Indicadores para los bucles de los parametros
% ymax = 0; ymin = 0;
% Valores max y min de salida para envolvente

% m = 0;
% mmax= length(lambdaRef) * length(K) * length(Tau) * length(T);
% N° de funciones
% k = 1;
% kmax = length(y.time);     % o length(y.data)
% N° de pasos o instantes muestreados

%
% lambdas = [0.246383536512386 0.512123142175763];
% lambdaRef=0.5;
% lambdaDis=0.5;
% lambdas = [lambdaRef lambdaDis];
% lambdas =  [0.6340 1.2928];
% Opción escogida por el AG
%
%
% J = TsgaSISO(lambdas)
J = FobjRSOTV([0.5 0.5])

% %% Peor caso de Total Variation
% TVWCmin = min(TVdata);
% TVWCmax = max(TVdata);
% TVWC(1,:) = [TVWCmax TVWCmin]
%
% %% Peor caso Tiempo de Establecimiento
% TestWCmax = max(Testdata);
% TestWCmin = min(Testdata);
% TestWC(1,:) = [TestWCmax TestWCmin]
%
% %% Peor caso Sobreoscilacion
% SOWCmax = max(SOdata(:,1));
% SOWCmin = min(SOdata(:,2));
% SOWC(1,:) = [SOWCmax SOWCmin]

%% Extremos de la envolvente
Ymax(:,1) = max(Y');
Ymin(:,1) = min(Y');
```

```
%% Ploteamos la envolvente
% figure
hold on
plot(t,Ymax,'b-')
hold on
plot(t,Ymin,'b-')
plot([1:1:length(Y)],yRP,'r--')
% % plot(Ynom.data, 'g-')

%% Comprobación extremos
% K = [1 3];
% Tau = [7 12];
% T = [1 3.5];

%% Comprobación de todos posibles casos
K = 1:1:3;
Tau = 7:1:12;
T = 1:0.5:3.5;

%% Casos aleatorios
% K = rand()*2+1;
% Tau = rand()*5+7;
% T = rand()*2.5+1;

% Ejecución función objetivo
% J = TsgaSISO([1.01908837619881 1.49917750264245])        % minimo
sin restricciones
J = FobjRSOTV([0.5173 0])                                  % mínimo con
restricciones

dataJ(base,1) = J;                                        % Base de
datos de la función objetivo
% hold on
% plot(t,Y,'r-','linewidth',1)

% %% Ploteamos la envolvente
% figure
hold on
plot(t,Ymax,'k','linewidth',1.5)
hold on
plot(t,Ymin,'k','linewidth',1.5)
plot([1:1:length(Y)],yRP,'r--')
% % plot(Ynom.data, 'g-')

%% Algoritmo Genético Matlab
% [x,fval,exitflag,output,population,scores] =
ga(fitnessfcn,nvars,...)
% InitialPopulationMatrix = 2;
% A = [1 0; -1 0; 0 1; 0 -1];
% b = [1; 0; 1; 0];
% [x,fval,exitflag,output,population,scores] = ga(@Tsfunga, 2, A,b)
% lb = [];
% ub = [];

%% Parámetros Algoritmo Genético Departamento (BasicGA)
% gaDat.Objfun='TsgaSISO';                % Función objetivo
% lb=[0 0];                               % Límites
% ub=[3 3];
% gaDat.FieldD=[lb; ub];
```

```
% gaDat.NIND=50;                                % N° individuos
% % gaDat.NIND=10;
% gaDat=ga(gaDat);                              % Ejecución del GA

    %% Tiempo de coste computacional
    cputsfunga = cputime-cputsfunga1;
    datatemp(base,:) = cputsfunga;
    %mean(datatemp)

% datagax(base,:) = gaDat.xmin
% datagaf(base,:) = gaDat.fxmin
base = base +1;
end
 w = warning('query','last');id = w.identifier;warning('on',id);

 mean(dataJ)
 max(dataJ)
 min(dataJ)
 dataPruebaJ = find(dataJ > 44.8125);
```

F. Código función objetivo de control DMC en SISO

El mismo que en el caso de QDMC.

G. Código del algoritmo de control DMC de un SISO

El mismo que en el caso de QDMC pero con menos parámetros (se eliminan las restricciones).

H. Código función objetivo de control QDMC en SISO

```
function [J] = FobjR(lambdas)

global c
global lambda
global alfa
global futref
global limits
global qdmc
% Parámetros referencias
global Ts
% Tiempo muestreo
global r
% Referencia
global t
% Tiempo simulación
% Parámetros modelo
```

```
global s
global K
global T
global Tau
% Filtros
global filtergu
global filterdgu
% Modelo nominal
global Tau0
global T0
global G0
global G
% Respuesta a escalón
global Su
global Sd
% Bases de datos
global TPdata
global datalambdas
global TVdata
global SOdata
global Testdata
global Y
global yRP
global k

%% Definición de lambdas
lambdaDis=lambdas(1);
lambdaRef=lambdas(2);

%% Indicadores para los bucles de los parametros
indK=1; indTau=1; indT=1;
m = 0;
mmax= length(lambdaRef) * length(K) * length(Tau) * length(T);
% N° de funciones
k = 1;
%kmax = length(y.time);      % o length(y.data)
% N° de pasos o instantes muestreados
kmax = length(r);
SOlimit = 0.98384;
TVlimit = 1.5;

datalambdas(k,:) = lambdas;
k = k+1;

%Definimos el modelo de los filtros
%Reference Tracking Filter
filtergu=1/(((Tau0*lambdaRef)*s+1));% fdt y/u
filtergu.OutputDelay = T0;
%Disturbancce Rejection Filter
filterdgu=1/(((Tau0*lambdaDis)*s+1));% fdt y/u
filterdgu.OutputDelay = T0;

while indK < length(K) + 1
    indTau = 1;

    while indTau < length(Tau) + 1
        indT = 1;

        while indT < length(T) + 1
            % Actualizo modelos con los datos de cada caso
```

```
            G = K(indK)/((Tau(indTau)*s+1));
% Modelo
            G.OutputDelay = T(indT);
% Retardo

            qdmc=qdmcinit(Su,Sd,c,lambda,alfa,futref,0,limits);
            %Simulamos el bucle de control con el DMC
            sim('qdmctfmcontrol')
            %Representamos el resultado
%             dibuja(Simul,'r','')


            %step(TP)
            m = m+1;

            %% Salida Y
            Y(m,:) = y.data;
            tY(m,:) = y.time;
%             stepdata = stepinfo(TP);
%             TSstepdata(m,:) = stepdata.SettlingTime;
%             OSstepdata(m,:) = stepdata.Overshoot;

%             %% lsiminfo
%             %              lsim(TP,r,t,'y-')
%             %               hold on
%             %               lsim(TotC,Freferencia,t,'b--')
%             [param, t] = lsim(TotC,Freferencia,t);
%             %              [param, t] = step(TP);
%             Y(:,m) = param;
%             PARAM = lsiminfo(param,t);
%             %SOSC = Y(PARAM.Max) - yRP;
%

            %% Cálculo de posición Regimen Permanente
            yRP = round(y.data(end));
            posRPprev = find(Y(m,:) < 0.98*yRP | Y(m,:) > 1.02*yRP);
% Posición en la que se supera el % de la franja

            %% Tiempo Establecimiento base de datos
            %posTest = find(Y(m,:) > 0.98*yRP & Y(m,:) < 1.02*yRP);
            if (posRPprev(end) == size(Y,2))
                posRP(m,:) = posRPprev(end);
            else
                posRP(m,:) = posRPprev(end)+1;
            end
            Testdata(m,:) = t(posRP(m));

            %% Sobreoscilación peores casos matricialmente
            %SOmaxm = max(y.data)
            posSO = find(Y(m,:) > yRP);
            if isempty(posSO)
                SOdata(m,1) = 0;
                SOdata(m,2) = 0;
            end
            if posSO ~= 0
                SOdata(m,1) = max(Y(m,posSO:kmax) - yRP);
                SOdata(m,2) = max(yRP - Y(m,posSO:kmax));
            end
%
```

```
%             TESTAB = PARAM.SettlingTime;
%             Testdata(m,:) = TESTAB;
%
%             % Constante de tiempo (0.1 es el intervalo transcurrido
entre dos medidas,
%             % se le resta 1, porque los indices empiezan en 1):
%             % tauEstim = 0.1*(posTest(1) - 1); % o t(posTest(1))
%
            %% Variación Total base de datos

            TVdata(m,:) = sum(Y(m,1:posRP(m)))-
sum(Y(m,1:posRPprev(end)));

            indT = indT + 1;
        end
        indTau = indTau + 1;
    end
    indK = indK + 1;
end

% SettlingTime = 'SettlingTime';
% Overshoot = 'Overshoot';
% TotalVariation = 'TotalVariation';
% TPdata =
struct(SettlingTime,TSstepdata,Overshoot,OSstepdata,TotalVariation,TVd
ata);

%% Peor caso Sobreoscilacion
SOWCsup = max(SOdata(:,1));
SOWCinf = max(SOdata(:,2));
SOWC = max(SOdata(:,:));

%% Solución J para TS
TS = min(Testdata);
J = TS;

%% Parámetros
violation = 0;
offset = 1e6;
SO = max(SOWC);
TV = max(TVdata);

% con violation SO y TV
if SO > SOlimit
    violation = (SO - SOlimit)/SOlimit;
    J = J+violation*offset;
end
if TV > TVlimit
    violation = (TV - TVlimit)/TVlimit;
    J = J+violation*offset;
end

%% con offset
% if SO > 0.3
%     J = J+offset
% end
% if TV > 100
%     J = J+offset
% end
```

```
end
```

I. Código del algoritmo de control QDMC de un SISO

```
% clear all
% close all

% w = warning('query','last');id = w.identifier;warning('off',id);
%w.identifier = Control:ltiobject:UseSSforInternalDelay;
base = 1;
while base < 2
    % tiempo de cpu invertido en la ejecucion del mencionado conjunto
de operaciones
    cputsfunga1 = cputime;        % tiempo de cpu invertido en la
ejecucion del mencionado conjunto de operaciones

    global c
    global lambda
    global alfa
    global futref
    global limits
    global qdmc
    % Parámetros referencias
    global Ts
% Tiempo muestreo
    global r
% Referencia
    global t
% Tiempo simulación
    % Parámetros modelo
    global s
    global K
    global T
    global Tau
    % Filtros
    global filtergu
    global filterdgu
    % Modelo nominal
    global Tau0
    global T0
    global G0
    global G
    % Respuesta a escalón
    global Su
    global Sd
    % Bases de datos
    global TPdata
    global datalambdas
    global TVdata
    global SOdata
    global Testdata
    global Y
    global yRP
    global k

    %definición inicial de función de transferencia
```

```
    s=tf('s');

    %% Definimos los parámetros de simulación
    steptimeDV=0;valueDV=0; %Escalón en la perturbación medible
    steptimeUDV=0;valueUDV=0; %Escalón en la perturbación medible
    outputnoisegain=0;%el ruido está entre +-outputnoisegain
    inputnoisegain=0;%el ruido está entre +-inputnoisegain
    Ts=8; % Tiempo muestreo
    r=[ones(1520/Ts,1)]; % Trayectoria de la referencia
    t=(0:Ts:Ts*size(r,1)-Ts)';% Tiempo para la trayectoria de la
referencia
    Tstop=t(end); %El tiempo de simulación lo fija la referencia que
definamos

    %% Etiqueta estructura modelos
    Kname = 'K';
    Tname = 'T';
    Tauname = 'Tau';
    LDname = 'LD';
    LRname = 'LR';

    %% Modelo extremos 1
%     K = [0.017404 0.028396];
%     Tau = [47.88 78.12];
%     T = [45.6 74.4];
%     LR = 1;
%     LD=2;
%     model(1) =
struct(Kname,K,Tauname,Tau,Tname,T,LDname,LD,LRname,LR);


%     % %% Modelo extremos 2
%     K = [0.017404 0.028396];
%     Tau = [11.4 18.6];
%     T = [45.6 74.4];
%     LR = 1.5;
%     LD = 1.5;
%     model(2) =
struct(Kname,K,Tauname,Tau,Tname,T,LDname,LD,LRname,LR);

    % %% Modelo extremos 3
    % K = [0.017404 0.028396];
    % Tau = [47.88 78.12];
    % T = [45.6 74.4];
    % LR = 2;
    % LD = 1;
    % model(3) =
struct(Kname,K,Tauname,Tau,Tname,T,LDname,LD,LRname,LR);

    %% Modelo extremos 4
    K = [0.017404 0.028396];
    Tau = [121.6 198.4];
    T = [47.8 78.12];
    LR = 2;
    LD = 1;
    model(4) =
struct(Kname,K,Tauname,Tau,Tname,T,LDname,LD,LRname,LR);

    %% Modelo Nominal
    % K0, Tau0, T0 son los valores nominales del proceso
```

```
    K0 = mean(K)+mean(K)/7;
    Tau0 = mean(Tau)+mean(Tau)/7;
    T0 = mean(T)+mean(T)/7;
    G0 = K0/((Tau0*s+1));
% fdt y/u proceso nominal
    G0.OutputDelay = T0;
    gdv=tf(0);% fdt y/dv (perturbación medible)
    gudv=tf(0);% fdt y/udv (perturbación no medible)


    %% Se calcula la respuesta del proceso para el horizon de
predicción definido
    n=360/Ts;% Orden del modelo
    Tstep=(n-1)*Ts;%Horizonte de predicción en segundos
    [Su,Tesc]=step(G0,(0:Ts:Tstep));%Su recoge la respuesta ante
escalón en u
%     figure
%     subplot(211)
%     plot(Tesc,Su,'-o')
%     title('Respuesta ante escalón en u')
%     xlabel('t(seg)')
%     ylabel('y')
    [Sd,Tesc]=step(gdv,(0:Ts:Tstep));%Sd recoge la respuesta ante
escalón en u
%     subplot(212)
%     plot(Tesc,Sd,'-o')
%     xlabel('t(seg)')
%     ylabel('y')
%     title('Respuesta ante escalón en dv')

    %% Ejemplo
    % figure
    %Parámetros del controlador QDMC
    % c: horizonte de control
    % alfa: penaliza error
    % lambda: penaliza acción de control
    c=2; lambda=0.001; alfa=1000000; futref=0;
    limits.umin=[-150];
    limits.umax=[150];
    limits.deltaumin=[-150];
    limits.deltaumax=[150];
    limits.ymin=[-100];
    limits.ymax=[100];
    limits.yminsoft=[-100];
    limits.ymaxsoft=[100];

    % qdmc=qdmcinit(Su,Sd,c,lambda,alfa,futref,0,limits);

        J = FobjR([2 1]);

        dataJ(base,1) = J;                                          % Base
de datos de la función objetivo

        %% Extremos de la envolvente
        Ymax(:,1) = max(Y);
        Ymin(:,1) = min(Y);

        %% Ploteamos la envolvente
        figure
        hold on
```

```
        plot(t,Ymax,'k','linewidth',1.5)
        hold on
        plot(t,Ymin,'k','linewidth',1.5)
        plot([1:1:t(end)],yRP,'r--')
        plot(t,Y,'m')
        % % plot(Ynom.data, 'g-')

%     %% Parámetros Algoritmo Genético Departamento (BasicGA)
%     gaDat.Objfun='FobjR';                % Función objetivo
%     lb=[0 0];                               % Límites
%     ub=[3 3];
%     gaDat.FieldD=[lb; ub];
%     %         gaDat.NIND=10;                          % N°
individuos
%     gaDat.NIND=50;
%     gaDat=ga(gaDat);                        % Ejecución del GA

    %%
    % [resultado,posicion] = min(gaDat.fxmin);
    % lambdas(iteracion) = gaDat.xmin(posicion);

    %% Tiempo de coste computacional
    cputsfunga = cputime-cputsfunga1;
    datatemp(base,:) = cputsfunga;
    %mean(datatemp)

%     datagax(base,:) = gaDat.xmin
%     datagaf(base,:) = gaDat.fxmin
    base = base +1;
end
% w = warning('query','last');id = w.identifier;warning('on',id);

mean(dataJ)
max(dataJ)
min(dataJ)
dataPruebaJ = find(dataJ > 44.8125);
```

J. Código función objetivo de control QDMC de un MIMO

```
function [J] = FobjMIMOR(lambdas)
global c
global lambda
global alfa
global futref
global limits
global qdmc
% Parámetros referencias
global Ts
% Tiempo muestreo
global r
% Referencia
global t
% Tiempo simulación
% Parámetros modelo
global s
```

```
global K
global T
global Tau

global K11
global T11
global Tau11
global K22
global T22
global Tau22

% Filtros
global filtergu
global filterdgu
% Modelo nominal
global Tau0
global T0
global G0
global G
% Respuesta a escalón
global Su
global Sd
% Bases de datos
global TPdata
global datalambdas
global TVdata
global SOdata
global Testdata
global Y
global yRP
global k

%% Definición de lambdas
lambdaDis=lambdas(1);
lambdaRef=lambdas(2);

%% Indicadores para los bucles de los parametros
indK=1; indTau=1; indT=1;
m = 0;
mmax= length(K) * length(Tau) * length(T);                % N° de
funciones
k = 1;
%kmax = length(y.time);     % o length(y.data)
% N° de pasos o instantes muestreados
kmax = length(r);
SOlimit = 0.98384;
TVlimit = 1.5;
Yprevia = [];

datalambdas(k,:) = lambdas;
k = k+1;

%Definimos el modelo de los filtros
%Reference Tracking Filter
filtergu0=1/(((Tau0*lambdaRef)*s+1));% fdt y/u
filtergu0.OutputDelay = T0;
filtergu = filtergu0*diag([1 1]);
%Disturbancce Rejection Filter
filterdgu0=1/(((Tau0*lambdaDis)*s+1));% fdt y/u
filterdgu0.OutputDelay = T0;
```

```
filterdgu = filterdgu0*diag([1 1]);

while indK < length(K) + 1
    indTau = 1;

    while indTau < length(Tau) + 1
        indT = 1;

        while indT < length(T) + 1
            % Actualizo modelos con los datos de cada caso
            G11 = K11(indK)/(Tau11(indTau)*s+1);
% Modelo
            G11.OutputDelay = T11(indT);
% Retardo

            G12 = 0;
            G21 = 0;

            G22 = K22(indK)/(Tau22(indTau)*s+1);
% Modelo
            G22.OutputDelay = T22(indT);

            G=[G11 G12; G21 G22];
            %
mimo_qdmc=mimo_qdmcinit(Su,Sd,c,lambda,alfa,futref,0,limits);
            %Simulamos el bucle de control con el DMC
            sim('mimo_qdmctfmcontrol')
            %Representamos el resultado
            %dibuja_mimo(Simul, 'c=2,alfa=[1 1],lambda=[1 1]');


            %step(TP)
            m = m+1;
            yRP = r(end);
            %% Salida Y
            if isempty(Yprevia)
                        Y = y.Data(:,1:2);
            else
%                 Y(:,(2*m-1):(2*m)) = [Yprevia , y.Data(:,1:2)];
%                 clear Y
                Y = [Yprevia , y.Data(:,1:2)];
            end
            Yprevia = Y;
            posRPprev1 = find(Y(:,(2*m-1)) < 0.98*yRP | Y(:,(2*m-1)) >
1.02*yRP);          % Posición en la que se supera el % de la franja
            posRPprev2 = find(Y(:,(2*m)) < 0.98*yRP | Y(:,(2*m)) >
1.02*yRP);          % Posición en la que se supera el % de la franja

        tY(:,m) = y.time;
        %             stepdata = stepinfo(TP);
        %             TSstepdata(m,:) = stepdata.SettlingTime;
        %             OSstepdata(m,:) = stepdata.Overshoot;

        %             %% lsiminfo
        %             %             lsim(TP,r,t,'y-')
        %             %              hold on
        %             %              lsim(TotC,Freferencia,t,'b--')
        %             [param, t] = lsim(TotC,Freferencia,t);
        %             %             [param, t] = step(TP);
```

```
        %             Y(:,m) = param;
        %             PARAM = lsiminfo(param,t);
        %             %SOSC = Y(PARAM.Max) - yRP;
        %

        %% Cálculo de posición Regimen Permanente
%         yRP = r(end);
        %             yRP = round(y.data(end));
        %             posRPprev = find(Y(m,:) < 0.98*yRP | Y(m,:) >
1.02*yRP);          % Posición en la que se supera el % de la franja
        %
        %% Tiempo Establecimiento base de datos
        %posTest = find(Y(m,:) > 0.98*yRP & Y(m,:) < 1.02*yRP);
        if (posRPprev1(end) == size(Y,1))
            posRP1(m,:) = posRPprev1(end);
        else
            posRP1(m,:) = posRPprev1(end)+1;
        end
        Testdata1(m,:) = t(posRP1(m));

        if (posRPprev2(end) == size(Y,1))
            posRP2(m,:) = posRPprev2(end);
        else
            posRP2(m,:) = posRPprev2(end)+1;
        end
        Testdata2(m,:) = t(posRP2(m));
%         Testdata(:,2*m-1:2*m) = [Testdata1, Testdata2]
        %% Sobreoscilación peores casos matricialmente
        %SOmaxm = max(y.data)
        %             posSO1 = find(Y((2*m-1),:) > yRP);
        %             if isempty(posSO1)
        %                 SOdata1(m,1) = 0;
        %                 SOdata1(m,2) = 0;
        %             end
        %             if posSO ~= 0
        %                 SOdata1(m,1) = max(Y(m,posSO1:kmax) - yRP);
        %                 SOdata1(m,2) = max(yRP - Y(m,posSO1:kmax));
        %             end

        %             posSO2 = find(Y(m,:) > yRP);
        %             if isempty(posSO2)
        %                 SOdata2(m,1) = 0;
        %                 SOdata2(m,2) = 0;
        %             end
        %             if posSO2 ~= 0
        %                 SOdata2(m,1) = max(Y(m,posSO2:kmax) - yRP);
        %                 SOdata2(m,2) = max(yRP - Y(m,posSO2:kmax));
        %             end

        %             TESTAB = PARAM.SettlingTime;
        %             Testdata(m,:) = TESTAB;
        %
        %             % Constante de tiempo (0.1 es el intervalo
transcurrido entre dos medidas,
        %             % se le resta 1, porque los indices empiezan en
1):
        %             % tauEstim = 0.1*(posTest(1) - 1); % o
t(posTest(1))
        %
        %% Variación Total base de datos
```

```
        %             TVdata(m,:) = sum(Y(m,1:posRP(m)))-
sum(Y(m,1:posRPprev(end)));
        %
        indT = indT + 1;
    end
    indTau = indTau + 1;
end
indK = indK + 1;
end

% SettlingTime = 'SettlingTime';
% Overshoot = 'Overshoot';
% TotalVariation = 'TotalVariation';
% TPdata =
struct(SettlingTime,TSstepdata,Overshoot,OSstepdata,TotalVariation,TVd
ata);

%% Peor caso Sobreoscilacion
% SOWCsup = max(SOdata(:,1));
% SOWCinf = max(SOdata(:,2));
% SOWC = max(SOdata(:,:));

%% Solución J para TS
TS1 = min(Testdata1);
TS2 = min(Testdata2);
J = TS1+TS2;
%
% %% Parámetros
% violation = 0;
% offset = 1e6;
% SO = max(SOWC);
% TV = max(TVdata);
%
% con violation SO y TV
% if SO > SOlimit
%     violation = (SO - SOlimit)/SOlimit;
%     J = J+violation*offset;
% end
% if TV > TVlimit
%     violation = (TV - TVlimit)/TVlimit;
%     J = J+violation*offset;
% end

%% con offset
% if SO > 0.3
%     J = J+offset
% end
% if TV > 100
%     J = J+offset
% end

end
```

K. Código del algoritmo de control QDMC de un MIMO

```
clear all
close all

% w = warning('query','last');id = w.identifier;warning('off',id);
%w.identifier = Control:ltiobject:UseSSforInternalDelay;
base = 1;
while base < 2
    % tiempo de cpu invertido en la ejecucion del mencionado conjunto
de operaciones
    cputsfunga1 = cputime;        % tiempo de cpu invertido en la
ejecucion del mencionado conjunto de operaciones

    global c
    global lambda
    global alfa
    global futref
    global limits
    % Parámetros referencias
    global Ts
% Tiempo muestreo
    global r
% Referencia
    global t
% Tiempo simulación
    % Parámetros modelo
    global s
    global K
    global T
    global Tau

    global K11
    global T11
    global Tau11
    global K22
    global T22
    global Tau22

    % Filtros
    global filtergu
    global filterdgu
    % Modelo nominal
    global Tau0
    global T0
    global G0
    global G
    % Respuesta a escalón
    global Su
    global Sd
    % Bases de datos
    global TPdata
    global datalambdas
    global TVdata
    global SOdata
    global Testdata
    global Y
    global yRP
    global k
```

```
    %definición inicial de función de transferencia
    s=tf('s');

    %% Definimos los parámetros de simulación
    Ts=8; % Tiempo muestreo
    steptimeDV=[0];valueDV=[0]; %Escalón en la perturbación medible (1
x num_dv)
    steptimeUDV=[0];valueUDV=[0]; %Escalón en la perturbación no
medible (1 x num_udv)
    outputnoisegain=0;%el ruido está entre +-outputnoisegain
    inputnoisegain=0;%el ruido está entre +-inputnoisegain
%     r1=[zeros(8/Ts,1);ones(1512/Ts,1)];% Trayectoria de la
referencia salida 1
%     r2=[zeros(440/Ts,1);ones(1080/Ts,1)];% Trayectoria de la
referencia salida 2
    r1=[zeros(8/Ts,1);ones(2392/Ts,1)];% Trayectoria de la referencia
salida 1
    r2=[zeros(440/Ts,1);ones(1960/Ts,1)];% Trayectoria de la
referencia salida 2
    r=[r1 r2]; % Trayectoria de las referencias (? x num_cv)
    t=(0:Ts:Ts*size(r,1)-Ts)';% Tiempo para la trayectoria de la
referencia
    Tstop=t(end); %El tiempo de simulación lo fija la referencia que
definamos

    %% Etiqueta estructura modelos
    Kname = 'K';
    Tname = 'T';
    Tauname = 'Tau';
    LDname = 'LD';
    LRname = 'LR';

    %% Modelo extremos 1
    K11 = [0.017404 0.028396];
    Tau11 = [47.88 78.12];
    T11 = [45.6 74.4];
    LR = 1;
    LD=2;
    model(1) =
struct(Kname,K,Tauname,Tau,Tname,T,LDname,LD,LRname,LR);

    %% Modelo extremos 2
    K = [0.017404 0.028396];
    Tau = [11.4 18.6];
    T = [45.6 74.4];
    LR = 1.5;
    LD = 1.5;
    model(2) =
struct(Kname,K,Tauname,Tau,Tname,T,LDname,LD,LRname,LR);

    %% Modelo extremos 3
    K = [0.017404 0.028396];
    Tau = [47.88 78.12];
    T = [45.6 74.4];
    LR = 2;
    LD = 1;
    model(3) =
struct(Kname,K,Tauname,Tau,Tname,T,LDname,LD,LRname,LR);
```

```
    %% Modelo extremos 4
    K22 = [0.017404 0.028396];
    Tau22 = [121.6 198.4];
    T22 = [47.8 78.12];
    LR = 2;
    LD = 1;
    model(4) =
struct(Kname,K,Tauname,Tau,Tname,T,LDname,LD,LRname,LR);

    %% Modelo Nominal
    % K0, Tau0, T0 son los valores nominales del proceso
    K0 = mean(K11);
    Tau0 = mean(Tau11);
    T0 = mean(T11);
    G011 = K0/((Tau0*s+1));
% fdt y/u proceso nominal
    G011.OutputDelay = T0;

    G012=0;
    G021=0;

    K0 = mean(K22);
    Tau0 = mean(Tau22);
    T0 = mean(T22);

    G022 = K0/((Tau0*s+1));
% fdt y/u proceso nominal
    G022.OutputDelay = T0;
    gu=[G011 G012; G021 G022]; % matriz fdt y/u
    % perturbaciones medibles
    gdv11=tf(0);
    gdv21=tf(0);
    gdv=[gdv11; gdv21]; % matriz fdt y/dv
    % perturbaciones no medibles
    gudv11=tf(0);
    gudv21=tf(0);
    gudv=[gudv11; gudv21]; % matriz fdt y/udv

    %% Calculamos coeficientes respuesta escalón
    [Su,Sd]=build_stepmodel(gu,gdv,Ts);

    %% Ejemplo
%     figure
    %Parámetros del controlador QDMC MIMO
    c=2;
    alfa=[10000 1000];
    lambda=[0.001 0.001];
    limits.deltaumin=[-1 -100];
    limits.deltaumax=[100 100];
    limits.umin=[-200 -200];
    limits.umax=[60  60];
    limits.ymin=[-100 -100];
    limits.ymax=[100 100];

    check_param; %chequea que las variables estén bien
    if qdmcerror
        disp('Error en la definición de los parámetros.')
    else
        %Simulamos el bucle de control con el QDMC
        J = FobjMIMOR([1.0748 1.4363]);
```

```
        %end


        % qdmc=qdmcinit(Su,Sd,c,lambda,alfa,futref,0,limits);

        %     dataJ(base,1) = J;
% Base de datos de la función objetivo
        %
            %% Extremos de la envolvente
            Ymax(:,1) = max(Y');
            Ymin(:,1) = min(Y');

            %% Ploteamos la envolvente
            figure
        %     hold on
            plot(t,Ymax,'k','linewidth',1.5)
            hold on
            plot(t,Ymin,'k','linewidth',1.5)
            plot([1:1:t(end)],yRP,'r--')
            plot(t,Y,'m')
            % % plot(Ynom.data, 'g-')

         %% Parámetros Algoritmo Genético Departamento (BasicGA)
%         gaDat.Objfun='FobjMIMOR';                % Función objetivo
%         lb=[0.1 0.1];                               % Límites
%         ub=[3 3];
%         gaDat.FieldD=[lb; ub];
%         gaDat.NIND=50;                           % N° individuos
%         %     gaDat.NIND=10;
%         gaDat=ga(gaDat);                         % Ejecución del GA

    end %el del else
    %%
    % [resultado,posicion] = min(gaDat.fxmin);
    % lambdas(iteracion) = gaDat.xmin(posicion);

    %% Tiempo de coste computacional
    cputsfunga = cputime-cputsfunga1;
    datatemp(base,:) = cputsfunga;
    %mean(datatemp)

%          datagax(base,:) = gaDat.xmin;
%          datagaf(base,:) = gaDat.fxmin;
    base = base +1;
end
% w = warning('query','last');id = w.identifier;warning('on',id);
```